Project 2: Perceptron Branch Predictor

Tai Duc Nguyen - ECEC412

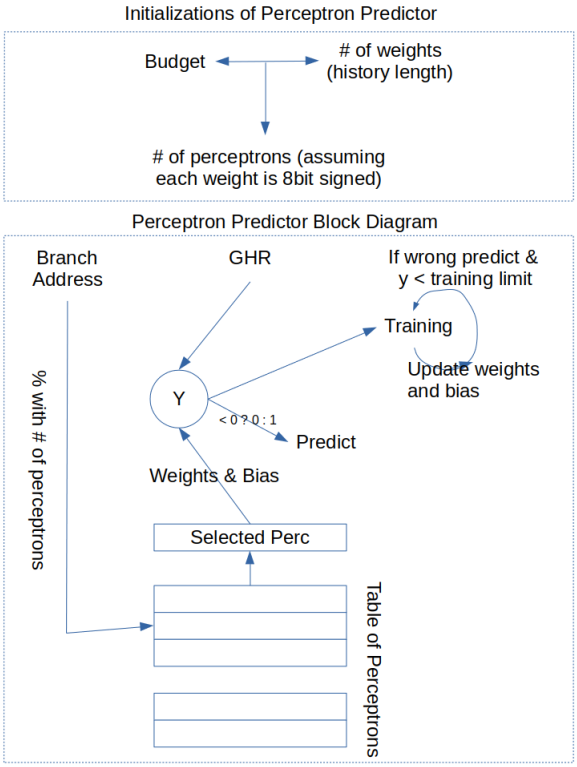
October 24, 2019

1 Introduction

Branch Prediction has been a serious research topic due to the importance of improving softwares' runtime. Various dynamic branch predictors had been proposed and brought amazing results. Common algorithms such as Bi-modal, Tournament and Gshare has been implemented and evaluated in Project 1. However, all of these predictors are very subceptible to *alaising* – when two unrelated branches destructively interfere by using the same prediction resources. Hence, this project introduces a Perceptron branch predictor based on the paper written by JimCnez and Lin: Dynamic Branch Prediction with Perceptrons (ISBN: 0-7695-1019-1/IEEE), which uses the simplest perceptron neural network of 1 input layer and 1 output layer. The algorithm and pseudo-code are described in the next Section.

2 Algorithm of the Perceptron Branch Predictor

The Perceptron Branch Predictor algorithm can be summed up with the following block diagram:



And the pseudo-code for this algorithm can be summarized below:

```
Initialize Predictor:
        perc_size = (n_weights + 1)*(bits_in_weight)
        n_perc = buget/perc_size

        for i=0 to n_perc-1
                perc_table[i].bias = 0
                for j=0 to n_weights-1
                        perc_table[i].weights[j] = 0
                end for
        end for

        GHR = 0

//-------------------------------------------------------------------//
Perceptron Training and Predicting:
        perc_idx = branch_addr % n_perc

        y = calculate_y(perc_table[perc_idx], GHR)
        predict = y >= 0 ? 1 : 0

        if (predict != outcome && abs(y) < training_limit) {
                update_weights()
                        for i=0 to n_weights-1
                                if (weights in weight_limits) { //(
                                    prevent overflow)
                                        if (GHR[i] == outcome) {
                                                perc_table[perc_idx].
                                                    weights[i] += 1
                                        } else {
                                                perc_table[perc_idx].
                                                    weights[i] -= 1
                                        }
                                }
                        end for
                end update_weights()
                update_bias()
                        if (perc_table[perc_idx].bias in weight_limits)
                            {
                                perc_table[perc_idx].bias += (outcome ?
                                    1 : -1)
                        }
                end update_bias()
        }

        shift_GHR()
        return predict == outcome
```

3 Evaluation of the Perceptron Branch Predictor

This branch predictor performs quite well in the 3 traces: 531.deepsjeng_r_branches, 541.leela_r_branches, and 548.exchange2_r_branches. The results are shown in the tables below:

531.deepsjeng_r_branches.cpu_trace					
BP Name	Buget(KB), # of weights	# of instructions	Correct predictions	Incorrect predictions	Accuracy (%)
perceptron	2, 22	204719966	144100480	60619486	70.39%
perceptron	4, 28	204719966	153345861	51374105	74.91%
perceptron	8, 34	204719966	161152979	43566987	78.72%
perceptron	16, 36	204719966	171983586	32736380	84.01%
perceptron	32, 59	204719966	173124866	31595100	84.57%
perceptron	64, 59	204719966	180136169	24583797	87.99%
perceptron	128, 62	204719966	183264056	21455910	89.52%
perceptron	256, 62	204719966	186367157	18352809	91.04%
perceptron	512, 62	204719966	187281412	17438554	91.48%
BP Name	Table size, Sat. counters	# of instructions	Correct predictions	Incorrect predictions	Accuracy (%)
gshare	2048, 1	204719966	163524096	41195870	79.88%
gshare	2048, 2	204719966	168679044	36040922	82.40%
gshare	4096, 2	204719966	174604586	30115380	85.29%
gshare	8192, 2	204719966	179529273	25190693	87.70%
gshare	16384, 2	204719966	183701357	21018609	89.73%
gshare	32768, 2	204719966	186896906	17823060	91.29%
gshare	65536, 2	204719966	189182509	15537457	92.41%

541.leela_r_branches.cpu_trace					
BP Name	Buget(KB), # of weights	# of instructions	Correct predictions	Incorrect predictions	Accuracy (%)
perceptron	2, 22	216132773	152451408	63681365	70.54%
perceptron	4, 28	216132773	153607259	62525514	71.07%
perceptron	8, 34	216132773	160809068	55323705	74.40%
perceptron	16, 36	216132773	165141436	50991337	76.41%
perceptron	32, 59	216132773	167986305	48146468	77.72%
perceptron	64, 59	216132773	172341832	43790941	79.74%
perceptron	128, 62	216132773	176481168	39651605	81.65%
perceptron	256, 62	216132773	178319134	37813639	82.50%
perceptron	512, 62	216132773	178622121	37510652	82.64%
BP Name	Table size, Sat. counters	# of instructions	Correct predictions	Incorrect predictions	Accuracy (%)
gshare	2048, 1	216132773	158111753	58021020	73.15%
gshare	2048, 2	216132773	163098178	53034595	75.46%
gshare	4096, 2	216132773	167678309	48454464	77.58%
gshare	8192, 2	216132773	172317632	43815141	79.73%
gshare	16384, 2	216132773	176292704	39840069	81.57%
gshare	32768, 2	216132773	179837616	36295157	83.21%
gshare	65536, 2	216132773	183053012	33079761	84.69%

548.exchange2_r_branches.cpu_trace					
BP Name	Buget(KB), # of weights	# of instructions	Correct predictions	Incorrect predictions	Accuracy (%)
perceptron	2, 22	353851200	277017174	76834026	78.29%
perceptron	4, 28	353851200	301868747	51982453	85.31%
perceptron	8, 34	353851200	319119944	34731256	90.18%
perceptron	16, 36	353851200	323973412	29877788	91.56%
perceptron	32, 59	353851200	321001679	32849521	90.72%
perceptron	64, 59	353851200	326665849	27185351	92.32%
perceptron	128, 62	353851200	326640147	27211053	92.31%
perceptron	256, 62	353851200	328956117	24895083	92.96%
perceptron	512, 62	353851200	329195735	24655465	93.03%
BP Name	Table size, Sat. counters	# of instructions	Correct predictions	Incorrect predictions	Accuracy (%)
gshare	2048, 1	353851200	306414277	47436923	86.59%
gshare	2048, 2	353851200	317228138	36623062	89.65%
gshare	4096, 2	353851200	324191040	29660160	91.62%
gshare	8192, 2	353851200	328448818	25402382	92.82%
gshare	16384, 2	353851200	331533004	22318196	93.69%
gshare	32768, 2	353851200	333952566	19898634	94.38%
gshare	65536, 2	353851200	336179077	17672123	95.01%

From this experiment's results, it is apparent that Gshare still out-performed the Perceptron predictor by a small margin at buget sizes bigger than 16k. This is possibly due to the fact that the indexing scheme for Gshare is much better in identifying different behavior of one branch since it take into account the value of the GHR during the Pattern History table-indexing process, while Perceptron only uses the branch address. Also, with branches whose behaviors out put a 1 for a long time, then switch to a 0 for another period, the weight vector needs to make, worst case, 128 + 1 miss predictions before it can make a correct one. In addition, since the neural network is so simple, with one input layer and one output layer, only linear relationship (AND, OR) between the branches can be learned. In contrast, Gshare can learn any Boolean relationships. However, the Perceptron algorithms should out-performed Gshare in applications where there are many more linear separable branches than inseparable ones.

The reason why the results above are different from the ones obtained by JimCnez and Lin is probably due to the different cpu trace used in the experiment (Figure 1 and 2)

4 Best configurations for the Perceptron Branch Predictor

The best configurations for this perdictor really come down to the available hardware budget. The highest jump in terms of accuracy results in going from 4KB to 8KB, or 8KB to 16KB. However, the higher the budget size, the better the performance of the predictor.

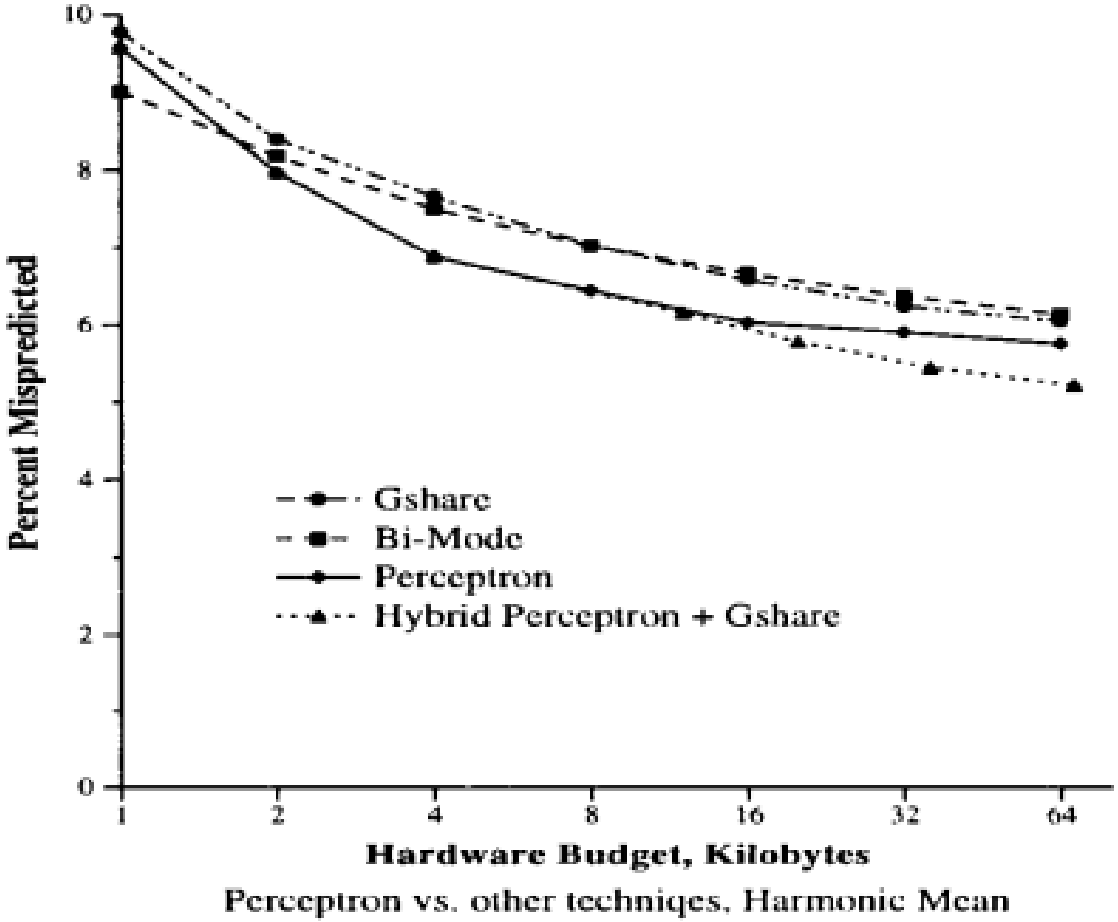


Figure 3: Hardware Budget vs. Prediction Rate on SPEC 2000. The perceptron predictor is more accurate than the two PHT methods at all hardware budgets over one kilobyte.

Figure 1: Accuracy curves reported by JimCnez and Lin

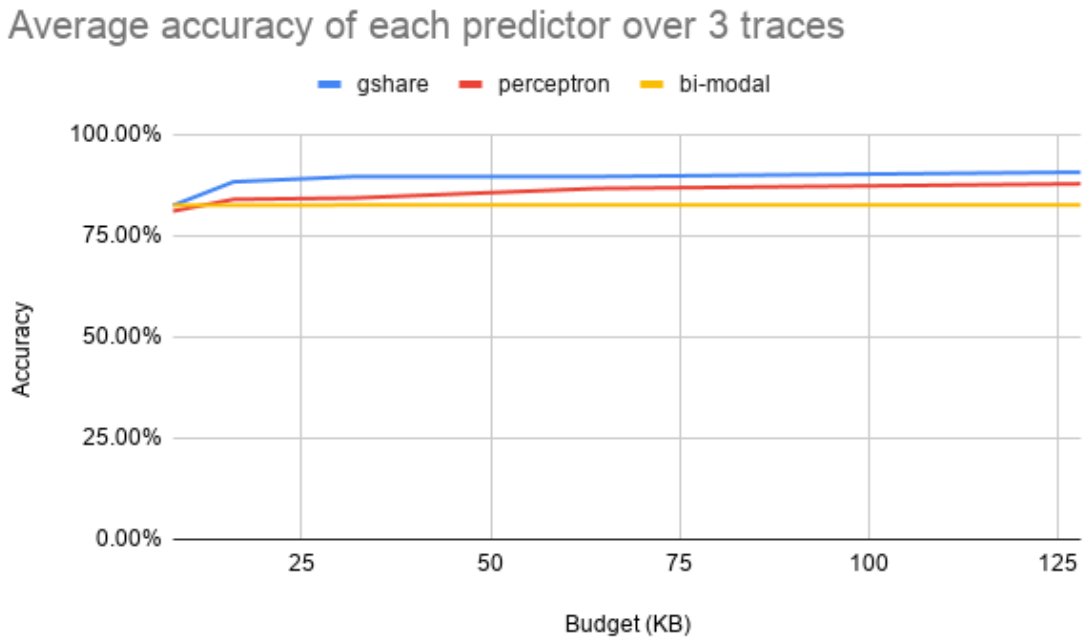


Figure 2: Accuracy curve from this experiment

5 Appendix

531.deepsjeng_r_branches.cpu_trace						
Predictor Type	Configuration	# Instructions	# Branches	Correct	Incorrect	Accuracy
twobitlocal	2048, 1	204719966	204719966	168274768	36445198	82.20%
twobitlocal	2048, 2	204719966	204719966	175794247	28925719	85.87%
twobitlocal	4096, 2	204719966	204719966	177385518	27334448	86.65%
twobitlocal	8192, 2	204719966	204719966	178085223	26634743	86.99%
twobitlocal	16384, 2	204719966	204719966	178223608	26496358	87.06%
twobitlocal	32768, 2	204719966	204719966	178223992	26495974	87.06%
twobitlocal	65536, 2	204719966	204719966	178224001	26495965	87.06%
tournament	2048, 8192, 8192	204719966	204719966	186733512	17986454	91.21%
tournament	4096, 8192, 8192	204719966	204719966	187322783	17397183	91.50%
tournament	4096, 16384, 16384	204719966	204719966	188849887	15870079	92.25%
gshare	2048, 1	204719966	204719966	163524096	41195870	79.88%
gshare	2048, 2	204719966	204719966	168679044	36040922	82.40%
gshare	4096, 2	204719966	204719966	174604586	30115380	85.29%
gshare	8192, 2	204719966	204719966	179529273	25190693	87.70%
gshare	16384, 2	204719966	204719966	183701357	21018609	89.73%
gshare	32768, 2	204719966	204719966	186896906	17823060	91.29%
gshare	65536, 2	204719966	204719966	189182509	15537457	92.41%

541.leela_r_branches.cpu_trace						
Predictor Type	Configuration	# Instructions	# Branches	Correct	Incorrect	Accuracy
twobitlocal	2048, 1	216132773	216132773	168602556	47530217	78.01%
twobitlocal	2048, 2	216132773	216132773	178713549	37419224	82.69%
twobitlocal	4096, 2	216132773	216132773	178989861	37142912	82.81%
twobitlocal	8192, 2	216132773	216132773	179391515	36741258	83.00%
twobitlocal	16384, 2	216132773	216132773	179415029	36717744	83.01%
twobitlocal	32768, 2	216132773	216132773	179417105	36715668	83.01%
twobitlocal	65536, 2	216132773	216132773	179417270	36715503	83.01%
tournament	2048, 8192, 8192	216132773	216132773	182428083	33704690	84.41%
tournament	4096, 8192, 8192	216132773	216132773	182744423	33388350	84.55%
tournament	4096, 16384, 16384	216132773	216132773	184227759	31905014	85.24%
gshare	2048, 1	216132773	216132773	158111753	58021020	73.15%
gshare	2048, 2	216132773	216132773	163098178	53034595	75.46%
gshare	4096, 2	216132773	216132773	167678309	48454464	77.58%
gshare	8192, 2	216132773	216132773	172317632	43815141	79.73%
gshare	16384, 2	216132773	216132773	176292704	39840069	81.57%
gshare	32768, 2	216132773	216132773	179837616	36295157	83.21%
gshare	65536, 2	216132773	216132773	183053012	33079761	84.69%

548.exchange2_r_branches.cpu_trace						
Predictor Type	Configuration	# Instructions	# Branches	Correct	Incorrect	Accuracy
twobitlocal	2048, 1	353851200	353851200	255602221	98248979	72.23%
twobitlocal	2048, 2	353851200	353851200	292073706	61777494	82.54%
twobitlocal	4096, 2	353851200	353851200	292146797	61704403	82.56%
twobitlocal	8192, 2	353851200	353851200	292178159	61673041	82.57%
twobitlocal	16384, 2	353851200	353851200	292195854	61655346	82.58%
twobitlocal	32768, 2	353851200	353851200	292198756	61652444	82.58%
twobitlocal	65536, 2	353851200	353851200	292200308	61650892	82.58%
tournament	2048, 8192, 8192	353851200	353851200	337742000	16109200	95.45%
tournament	4096, 8192, 8192	353851200	353851200	337826831	16024369	95.47%
tournament	4096, 16384, 16384	353851200	353851200	338578309	15272891	95.68%
gshare	2048, 1	353851200	353851200	306414277	47436923	86.59%
gshare	2048, 2	353851200	353851200	317228138	36623062	89.65%
gshare	4096, 2	353851200	353851200	324191040	29660160	91.62%
gshare	8192, 2	353851200	353851200	328448818	25402382	92.82%
gshare	16384, 2	353851200	353851200	331533004	22318196	93.69%
gshare	32768, 2	353851200	353851200	333952566	19898634	94.38%
gshare	65536, 2	353851200	353851200	336179077	17672123	95.01%