Dynamic Branch Prediction with Perceptrons

Daniel A. Jiménez, Calvin Lin
The University of Texas at Austin
HPCA 2001

Presented by Nils Wistoff
Mentors: Firtina Can, Salami Behzad, Hasan Hasan
Seminar in Computer Architecture
April 29, 2021
Outline

1. Branch Prediction
2. Paper Summary
   1. Executive Summary
   2. Background & Related Work
   3. Key Idea: Perceptron as Branch Predictor
   4. Implementation
   5. Results
   6. Optimization: Hybrid Predictor
   7. Final Results
   8. Conclusion
3. Analysis and Discussion
Processor Pipeline

IF | ID | Issue | EX | WB
---|----|-------|----|----

Dynamic Branch Prediction with Perceptrons - HPCA 2001

29.04.21
Processor Pipeline

Instructions enter through the processor frontend.
Processor Pipeline

Instructions enter through the processor frontend.
Processor Pipeline

Instructions enter through the processor frontend.
Processor Pipeline

Instructions are executed back-to-back
Processor Pipeline

Instructions are executed back-to-back.
Processor Pipeline

Instructions are executed back-to-back
Processor Pipeline

Goal: 1 instr/cycle
Control Hazard

IF | ID | Issue | EX | WB
---|----|-------|----|---

Branch A or B
Instr1

A2 | A1
B2 | B1

29.04.21
Dynamic Branch Prediction with Perceptrons - HPCA 2001
Control Hazard

[Diagram showing a control hazard flow with stages IF, ID, Issue, EX, and WB, and branches A or B and instructions Istr1]
Control Hazard

IF cannot know whether to continue at A or B
Control Hazard

IF cannot know whether to continue at A or B
Control Hazard

IF cannot know whether to continue at A or B
Control Hazard

IF | ID | Issue | EX | WB

Branch A or B | Instr1

Branch is resolved, result is sent to IF
## Control Hazard

<table>
<thead>
<tr>
<th>IF</th>
<th>ID</th>
<th>Issue</th>
<th>EX</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>A4</td>
<td>A3</td>
<td>A2</td>
<td>A1</td>
<td>Branch A or B</td>
</tr>
<tr>
<td>Frontend can continue fetching</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

29.04.21

Dynamic Branch Prediction with Perceptrons - HPCA 2001
Control Hazard

3 cycles lost!
Branch Prediction

Dynamic Branch Prediction with Perceptrons - HPCA 2001
## Branch Prediction

### Flowchart

**IF A?**

<table>
<thead>
<tr>
<th>A4</th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
</tr>
</thead>
</table>

- **Branch A or B**

**Instr1**

<table>
<thead>
<tr>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
</tr>
</thead>
</table>

*Speculatively fetch one branch*

### Table

<table>
<thead>
<tr>
<th>IF</th>
<th>ID</th>
<th>Issue</th>
<th>EX</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>A?</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A4</th>
<th>A3</th>
<th>A2</th>
<th>A1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B4</th>
<th>B3</th>
<th>B2</th>
<th>B1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

29.04.21 Dynamic Branch Prediction with Perceptrons - HPCA 2001
## Branch Prediction

<table>
<thead>
<tr>
<th>IF A?</th>
<th>ID</th>
<th>Issue</th>
<th>EX</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A1</td>
<td>Branch A or B</td>
<td>Instr1</td>
<td></td>
</tr>
</tbody>
</table>

Speculatively fetch one branch
### Branch Prediction

#### IF

<table>
<thead>
<tr>
<th>A6</th>
<th>A5</th>
<th>A4</th>
<th>A3</th>
</tr>
</thead>
</table>

#### ID

<table>
<thead>
<tr>
<th>A2</th>
<th>A1</th>
</tr>
</thead>
</table>

#### Issue

<table>
<thead>
<tr>
<th>Branch A or B</th>
</tr>
</thead>
</table>

#### EX

<table>
<thead>
<tr>
<th>Instr1</th>
</tr>
</thead>
</table>

#### WB

Speculatively fetch one branch

---

29.04.21

Dynamic Branch Prediction with Perceptrons - HPCA 2001
Branch Prediction

![Branch Prediction Diagram](image)

Branch is resolved
Branch Prediction

Correct Prediction: We won!
## Branch Prediction

<table>
<thead>
<tr>
<th>IF</th>
<th>ID</th>
<th>Issue</th>
<th>EX</th>
<th>WB</th>
</tr>
</thead>
<tbody>
<tr>
<td>B4</td>
<td>B3</td>
<td>B2</td>
<td>B1</td>
<td>B1</td>
</tr>
<tr>
<td>B4</td>
<td>B3</td>
<td>B2</td>
<td>B1</td>
<td>B1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Branch A or B</td>
<td>Instr1</td>
</tr>
</tbody>
</table>

**Misprediction:** We pay a 3 cycle penalty
The Goal of Branch Prediction

How to make a good prediction?

IF | ID | Issue | EX | WB
---|----|-------|----|---
Branch A or B | Instr1 |          |    |    

A4 | A3 | A2 | A1
B4 | B3 | B2 | B1

29.04.21
Dynamic Branch Prediction with Perceptrons - HPCA 2001
Outline

1. Branch Prediction

2. Paper Summary
   1. Executive Summary
   2. Background & Related Work
   3. Key Idea: Perceptron as Branch Predictor
   4. Implementation
   5. Results
   6. Optimization: Hybrid Predictor
   7. Final Results
   8. Conclusion

3. Analysis and Discussion
Executive Summary

Problem
Branch predictor’s **accuracy** is central for performance

Goal
Create a **branch predictor that is superior** to state of the art

Key Idea
**Perceptrons are efficient** in learning and predicting certain patterns over long histories

Mechanism
Build a branch predictor with **perceptrons as central decision makers**
Implement a hybrid perceptron / conventional predictor

Results
**Perceptron:** 14.7% **improvement** over baseline for a fixed 4K HW budget
**Hybrid Predictor:** **Outperforms baseline** consistently across benchmarks
Outline

1. Branch Prediction
2. Paper Summary
   1. Executive Summary
   2. Background & Related Work
   3. Key Idea: Perceptron as Branch Predictor
   4. Implementation
   5. Results
   6. Optimization: Hybrid Predictor
   7. Final Results
   8. Conclusion
3. Analysis and Discussion
Static Branch Prediction

• Idea: Branches are mostly biased
  • e.g. Backward Taken, Forward Not Taken
• Let compiler profile and arrange code

• Trivial to implement in HW
• Branch behaviour often not known at compile time
• Only efficient for heavily biased branches

li a0, 16
l1:
addi a0, -1
...
bnez a0, l1
...

6.25% Misprediction Rate
Branch History Table (BHT)

0  li    a0, 16
l1:  1  addi  a0, -1
...
  x  bneqz a0, l1
...

Prediction:
1. Index BHT with branch address
2. Take branch if MSB of BHT entry equals 1
Branch History Table (BHT)

0 li a0, 16
l1:
1 addi a0, -1
...
x bneqz a0, l1
...

Update (after branch is resolved):
1. Increment BHT entry if branch was taken, decrement otherwise
Branch History Table (BHT)

0  li       a0, 16  
  l1:  
1  addi    a0, -1  
...  
10  bneqz  a0, l1  
...

Prediction:
1. Index BHT with branch address
2. Take branch if MSB of BHT entry equals 1
Branch History Table (BHT)

0 li a0, 16
l1:  
1 addi a0, -1
...
  x bneqz a0, l1
...

Update (after branch is resolved):
1. Increment BHT entry if branch was taken, decrement otherwise
Branch History Table (BHT)

- Very Simple Design
- Not context-aware

```
0 li   a0, 16
l1:
1 addi a0, -1
...
X bneqz a0, l1
...
```

![BHT Diagram]

```
PC
Predict: Taken
```

Pattern History Table (PHT): *gshare*

**Global History Register**

| B₆ | B₅ | B₄ | B₃ | B₂ | B₁ |

**Program Counter (Lower Bits)**


**Problem:** PHT scales exponentially with history length.
Perceptron

• Takes set of *features*
• Makes *prediction* based on features (*Inference*)
• Improves over time (*training*)
Perceptron: Inference

\[ y = x \cdot w^T \]
Perceptron: Training

Training: (in case of misprediction) \( w' = w + y_0 \cdot x \)

Ground Truth

29.04.21 Dynamic Branch Prediction with Perceptrons - HPCA 2001
Outline

1. Branch Prediction
2. Paper Summary
   1. Executive Summary
   2. Background & Related Work
   3. Key Idea: Perceptron as Branch Predictor
   4. Implementation
   5. Results
   6. Optimization: Hybrid Predictor
   7. Final Results
   8. Conclusion
3. Analysis and Discussion
Perceptron as Branch Predictor

• One perceptron per branch location
• Use Global History Register as input
  • Entries are either -1 (not taken) or 1 (taken)
  • Constant 1 input for branch bias
• If $y \geq 0$ take branch, else do not take
• Once branch outcome is known, dynamically train in case of misprediction
Perceptron as Branch Predictor: Observations

• Perceptron output $y$ contains confidence about decision

• Each weight corresponds to an input and can be interpreted as follows:
  • $w_n >> 0$: Positive correlation to $-n$th branch outcome
  • $w_n << 0$: Negative correlation to $-n$th branch outcome
  • $w_n \approx 0$: Little correlation to $-n$th branch outcome

\[
y \geq 0 \quad \text{Take branch}
\]
\[
y < 0 \quad \text{Do not take branch}
\]
Outline

1. Branch Prediction
2. Paper Summary
   1. Executive Summary
   2. Background & Related Work
   3. Key Idea: Perceptron as Branch Predictor
   4. Implementation
   5. Results
   6. Optimization: Hybrid Predictor
   7. Final Results
   8. Conclusion
3. Analysis and Discussion
Putting it Together

1. **Hash branch address** to select perceptron.
2. **Fetch weights** of selected perceptron from SRAM.
3. **Compute** $y$ from weights and global history.
4. Take branch if $y > 0$.
5. **Train** perceptron after branch is resolved.
6. **Write back** perceptron.

Dynamic Branch Prediction with Perceptrons - HPCA 2001
Delay

- 1 cycle to access SRAM
- Approx. 2 cycles for inference
- Approx. **3 cycles** from branch address to prediction.
- We do not want to wait that long!
Pipelined Operation

1. When branch is encountered, use prediction from previous iteration.
2. After branch is resolved, train and write back perceptron.
3. Update global history register, concatenate outcome with branch address, select next entry.
4. Fetch weights of selected perceptron from SRAM.
5. Make prediction for next branch.
Outline

1. Branch Prediction
2. Paper Summary
   1. Executive Summary
   2. Background & Related Work
   3. Key Idea: Perceptron as Branch Predictor
   4. Implementation
   5. Results
   6. Optimization: Hybrid Predictor
   7. Final Results
   8. Conclusion
3. Analysis and Discussion
Methodology

• Baseline: gshare, bi-mode (PHT predictors)
• Choose fixed HW budget (4KiB in the following)
• Generate traces of SPEC95 and SPEC2000 benchmarks
• Simulate branch predictors’ performance for benchmark traces
Misprediction Rates

Why does the perceptron perform well?

Baseline

Perceptron

...but not always

Perceptron often outperforms PHTs...
History Length: PHT

Global History Register

| B_6 | B_5 | B_4 | B_3 | B_2 | B_1 |

Program Counter (Lower Bits)


PHT

<table>
<thead>
<tr>
<th>...</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>00</td>
</tr>
<tr>
<td>...</td>
<td>10</td>
</tr>
<tr>
<td>...</td>
<td>01</td>
</tr>
<tr>
<td>...</td>
<td>11</td>
</tr>
<tr>
<td>...</td>
<td>10</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
History Length: PHT

Global History Register

| B_7 | B_6 | B_5 | B_4 | B_3 | B_2 | B_1 |

Program Counter (Lower Bits)


Adding one bit of history doubles PHT size

PHT

| ... | ... |
| ... | 00 |
| ... | 10 |
| ... | 01 |
| ... | 11 |
| ... | 10 |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
| ... | ... |
History Length: Perceptron

\[
\begin{pmatrix}
1 \\
B_{-1} \\
B_{-2} \\
B_{-3} \\
B_{-4} \\
B_{-5} \\
B_{-6}
\end{pmatrix}
\cdot
\begin{pmatrix}
w_0 \\
w_1 \\
w_2 \\
w_3 \\
w_4 \\
w_5 \\
w_6
\end{pmatrix}^T = y
\]
Adding one bit of history adds single weight.
History Length: Supporting Experiment

- Same HW budget for both predictors
- Measure performance over different history lengths
- Perceptron can score for large history lengths

![Graph showing percent mispredicted vs history length with Gshare and Perceptron lines]

- Gshare’s maximum @18
- Perceptron’s maximum @>60
Misprediction Rates

Why does the perceptron perform well?
→ *Efficiently captures long histories*

When does the perceptron perform well?

...but not always

Perceptron often outperforms PHTs...
Linear Separability

\( a, b \in \{0, 1\} \)

\[
\begin{array}{c|c|c}
B1: & T & N \\
\hline
B2: & T & T \\
B3: & T & T \\
\end{array}
\]

\[
\begin{array}{c|c|c}
B1: & T & N \\
\hline
B2: & T & N \\
B3: & T & T \\
\end{array}
\]

\[
\begin{pmatrix}
w_0 \\
w_1 \\
w_2 \\
\end{pmatrix} =
\begin{pmatrix}
1 \\
-1 \\
1 \\
\end{pmatrix}
\]

\[
\begin{array}{c|c|c}
B1: & T & N \\
\hline
B2: & T & N \\
B3: & T & N \\
\end{array}
\]

\[
\begin{pmatrix}
w_0 \\
w_1 \\
w_2 \\
\end{pmatrix} =
\begin{pmatrix}
? \\
? \\
? \\
\end{pmatrix}
\]
Linear Separability: Supporting Experiment

- Sort benchmarks by share of linearly separable branches
- Perceptron outperforms Gshare on benchmarks with many linearly separable branches
Misprediction Rates

Why does the perceptron perform well?
→ **Efficiently captures long histories**

When does the perceptron perform well?
→ **When branch function is linearly separable**
Outline

1. Branch Prediction
2. Paper Summary
   1. Executive Summary
   2. Background & Related Work
   3. Key Idea: Perceptron as Branch Predictor
   4. Implementation
   5. Results
   6. Optimization: Hybrid Predictor
   7. Final Results
   8. Conclusion
3. Analysis and Discussion
Hybrid Predictor

- Implement *both* perceptron and gshare
- Use a *choice table* to select predictor on branch
Outline

1. Branch Prediction

2. Paper Summary
   1. Executive Summary
   2. Background & Related Work
   3. Key Idea: Perceptron as Branch Predictor
   4. Implementation
   5. Results
   6. Optimization: Hybrid Predictor
   7. Final Results
   8. Conclusion

3. Analysis and Discussion
Misprediction Rates

Dynamic Branch Prediction with Perceptrons - HPCA 2001

Baseline
Hybrid

Hybrid predictor consistently outperforms PHTs
Outline

1. Branch Prediction

2. Paper Summary
   1. Executive Summary
   2. Background & Related Work
   3. Key Idea: Perceptron as Branch Predictor
   4. Implementation
   5. Results
   6. Optimization: Hybrid Predictor
   7. Final Results
   8. Conclusion

3. Analysis and Discussion
Executive Summary

Problem
Branch predictor’s accuracy is central for performance

Goal
Create a branch predictor that is superior to state of the art

Key Idea
Perceptrons are efficient in learning and predicting certain patterns over long histories

Mechanism
Build a branch predictor with perceptrons as central decision makers
Implement a hybrid perceptron / conventional predictor

Results
Perceptron: 14.7% improvement over baseline for a fixed 4K HW budget
Hybrid Predictor: Outperforms baseline consistently across benchmarks
Outline

1. Branch Prediction
2. Paper Summary
   1. Executive Summary
   2. Background & Related Work
   3. Key Idea: Perceptron as Branch Predictor
   4. Implementation
   5. Results
   6. Optimization: Hybrid Predictor
   7. Final Results
   8. Conclusion

3. Analysis and Discussion
Takeaways

• First dynamic branch predictor using a neural network
• Performs very well for class of linearly separable branches
• Hybrid predictor can efficiently combine benefits of PHT and perceptron
Strengths

• Simple, straight-forward idea
• Effective and efficient solution
• Detailed performance analysis and discussion with lots of supporting evidence
Weaknesses

• Predictor performance strongly depends on benchmark. Might not always be an improvement.

• High latency of perceptron

• Weak area analysis
  • “by examining die photos, we estimate [...]”

• Weak timing analysis
  • Estimate by comparing to multiplier
  • “we believe that a good implementation [...] will take no more than two clock cycles [...]”
Neural Net Prediction

Scary Smart Prediction

- A true artificial network inside every “Zen” processor
- Builds a model of the decisions driven by software code execution
- Anticipates future decisions, pre-load instructions, choose the best path through the CPU

Impact

• Perceptron branch predictor known to be used in several modern processors [1, 2]
• However, not many details are public
• This work is one pillar of modern branch prediction research [3]

Questions / Comments?
How could we improve the perceptron predictor so that it works for more types of branches?
Do you see further use cases for neural networks in processor microarchitecture?