CS 203 Project

Load Value Prediction

Github username: UCR-CS203/**final-project-advca_samvarsh**

INTRODUCTION

We all know about temporal and spatial locality. Assume a block is retrieved from memory into cache as we needed a word in that block. Temporal locality says that there is a high probability that we may need that word again in the near future. Spatial locality refers to the fact that it is highly likely that we may need the other data in the block. Similarly, value locality describes the likelihood of the recurrence of a previously seen value within a storage location. Here I exploit value locality to achieve load value prediction. I am following this paper, https://dl.acm.org/doi/pdf/10.1145/248209.237173.

APPROACH AND DESIGN

Value locality exploits the relationship between the instruction address and the value being loaded. I started with implementing the basic version by just having a **Load Value Prediction Table**(LVPT) and predicting the values based on the last loaded value by that particular instruction. First time when a new instruction comes in, it is simply added to the table along with its actual value. From the next time onwards, when an already seen instruction comes in, its value is compared with the last loaded vaue. If it turns out to be wrong the new value is added as the value for that instruction. This simple last value predictor was checked with three traces, and the respective accuracies are summarised in the table below.

Taking it to the next level

Along with the LVPT, another table was added – **Load Classification Table(LCT)**. This table holds the instruction address and set of history bits. This determines the confidence, that is, if an instruction is good enough to be predicted or not. Based on the confidence values, the instructions are classified as *unpredictable*, *predictable* and *constant* loads. A 2-bit counter is used with each of its values indicating the following – *"unpredictable", "unpredictable", "predictable", "constant".* Each time an instruction comes in, the counter is either incremented or decremented based on if the prediction is right or wrong.

EXPERIMENTAL SETUP

Three traces were used, each one with 5000, 200,000 and 600,000 instructions. First the base case , last value predictor was checked. Then the traces were tested on the predictor with LVPT and LCT. There the traces were tested in two different ways – predicting when the confidence is 25% and predicting when the confidence is 50%. The 50% case showed a better accuracy. The actual accuray values are discussed in the next section. Here by 50% and 25% accuracy I mean, having the first two states as unpredictable and just having the first stage as unpredictable in the 2-bit predictor.

EVALUATION AND ANALYSIS

Base Case Analysis – Prediction based on last stored value

```
Total number of insructions: 5000
Number of instructions predicted:4165
Number of correct predictions:1357
Number of instructions not predicted:835
C:\Users\varsh\UCR\CLASSES\CA\Project\BaseCase\final-project-advca_samvarsh-main\BasseCase\x64\Debug\BasseCase.exe (proc
ess 5724) exited with code 0.
Press any key to close this window . . .
```

```
Total number of insructions: 237199
Number of instructions predicted:233375
Number of correct predictions:102880
Number of instructions not predicted:3824
C:\Users\varsh\UCR\CLASSES\CA\Project\BaseCase\final-project-advca_samvarsh-main\BasseCase\x64\Debug\BasseCase.exe (proc
ess 38632) exited with code 0.
Press any key to close this window . . .
```

```
Total number of insructions: 618942
Number of instructions predicted:613054
Number of correct predictions:304207
Number of instructions not predicted:5888
C:\Users\varsh\UCR\CLASSES\CA\Project\BaseCase\final-project-advca_samvarsh-main\BasseCase\x64\Debug\BasseCase.exe (proc
ess 32108) exited with code 0.
Press any key to close this window . . .
```

Anaysis with LVPT and LCT tables

Predicting when I get 50% confidence

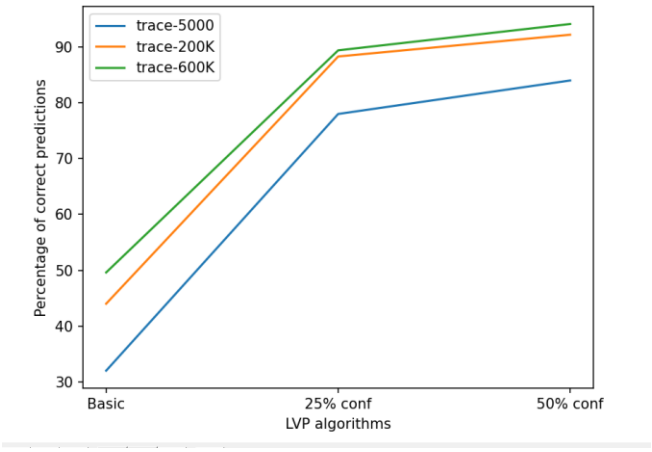
```
Total number of instructions: 5000
Number of instructions predicted:731
Number of correct predictions:620
No of instructions not predicted:4269
C:\Users\varsh\UCR\CLASSES\CA\Project\LoadValuePrediction\x64\Debug\LoadValuePrediction.exe (process 8060) exited with c
ode 0.
```

```
Total number of instructions: 237199
Number of instructions predicted:101209
Number of correct predictions:93362
No of instructions not predicted:135990
C:\Users\varsh\UCR\CLASSES\CA\Project\LoadValuePrediction\x64\Debug\LoadValuePrediction.exe (process 32876) exited with
code 0.
```

```
Total number of instructions: 618942
Number of instructions predicted:295312
Number of correct predictions:277945
No of instructions not predicted:323630
C:\Users\varsh\UCR\CLASSES\CA\Project\LoadValuePrediction\x64\Debug\LoadValuePrediction.exe (process 31528) exited with
code 0.
```

Summarising the above results

	Base Case	**LVPT and LCT (25% confidence)**	**LVPT and LCT (50% confidence)**
5K instructions	32%	78%	84.8%
200K instructions	44%	88.3%	92.2%
600K instructions	49.6%	89.4%	94.1%



We can see from the above graph that the prediction percentage has increased as we improve the algorithm from being just last value prediction to prediction based on the instructions's history. But the downside here is that on an average, as we see from the output screenshots attached, 50% of the instructions go as unpredictable. May be a learning based algorithm such as reinforcement learning which can learn and predict as the instructions come in may help to improve that.

If we consider that the system takes 150 cycles for a memory fetch (load), performing this prediction based on value locality yields a *speedup* of 2 in both the base case and the case with LCT. But the differnece is that, the second does very less wrong predictions compared the base case because of the use of history bits and confidence calculations.

Thus summarising, the predictor implemeted above classifies 50% of the instructions under the unpredictable category and 50% under the predictable category. Among the ones it predicts, it does it with a 90% accuracy.