Compiler-level Optimisation Branch Prediction

A mini-lecture series

CSE498 Collaborative Design - Secure and Efficient C++ Software Development

01/15/2025

Kira Chan

https://cse.msu.edu/~chanken1/

Mini lecture series

- Over the course of the semester
- One mini piece of information that I think is very useful
- Range from application design theory, programming, security, other programming concepts, etc.
- Also include a famous computer scientist and their contributions

Most upvoted stackoverflow question

- Take a long array of random items (say 100,000 items)
  - int random_array[] = {7, 116, 2, 236, 68, 70, 227, 75, 170, 119}
- random_array.sort()
- int sum
- for(auto& item: random_array) {
  - if(item > 128): sum += item;
- }

- https://stackoverflow.com/questions/11227809/why-is-processing-a-sorted-array-faster-than-processing-an-unsorted-array

Some things to consider

- Sorting is lower bounded by nlogn
- You must loop through the entire array

Why is processing a sorted array faster than processing an unsorted array?

Ask Question

Asked 12 years, 6 months ago Modified 29 days ago Viewed 1.9m times

27399

In this C++ code, sorting the data (*before* the timed region) makes the primary loop ~6x faster:

```
#include <algorithm>
#include <ctime>
#include <iostream>

int main()
{
    // Generate data
    const unsigned arraySize = 32768;
    int data[arraySize];

    for (unsigned c = 0; c < arraySize; ++c)
        data[c] = std::rand() % 256;

    // !!! With this, the next loop runs faster.
    std::sort(data, data + arraySize);

    // Test
    clock_t start = clock();
    long long sum = 0;
    for (unsigned i = 0; i < 100000; ++i)
    {
        for (unsigned c = 0; c < arraySize; ++c)
        {   // Primary loop.
            if (data[c] >= 128)
                sum += data[c];
        }
    }

    double elapsedTime = static_cast<double>(clock()-start) / CLOCKS_PER_SEC;

    std::cout << elapsedTime << '\n';
    std::cout << "sum = " << sum << '\n';
}
```

- Without `std::sort(data, data + arraySize);`, the code runs in 11.54 seconds.
- With the sorted data, the code runs in 1.93 seconds.

Branch prediction

35130

+2200

You are a victim of branch prediction fail.

What is Branch Prediction?

Consider a railroad junction:

Image by Mecanismo, via Wikimedia Commons. Used under the CC-By-SA 3.0 license.

Guessing

- If you guess right, then the train keeps going
- If you guess wrong, then the train must stop, backup, and restart

- Modern processors are slow and have long pipelines. This means they take forever to “warm up” and “slow down”

- Compiler is trying to identify a pattern and follow it

Results

- If the array is sorted, then the compiler essentially will be correctly guessing most of the time if it used the previous data
- If the array is not sorted, then it is random guessing

- Sorted

```
T = branch taken
N = branch not taken

data[] = 0, 1, 2, 3, 4, ... 126, 127, 128, 129, 130, ... 250, 251, 252, ...
branch = N  N  N  N  N  ...   N    N    T    T    T  ...   T    T    T  ...

       = NNNNNNNNNNNN ... NNNNNNNTTTTTTTTT ... TTTTTTTTT  (easy to predict)
```

- Not sorted

```
data[] = 226, 185, 125, 158, 198, 144, 217, 79, 202, 118,  14, 150, 177, 182, ...
branch =   T,   T,   N,   T,   T,   T,   T,  N,   T,   N,   N,   T,   T,   T  ...

       = TTNTTTTNTNNTTT ...   (completely random - impossible to predict)
```

Person of the day
Bjarne Stroustrup

- Developed the C++ language
- “C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do it blows your whole leg off.”