CSC 656-01, S23
Extra Credit Coding Project
Due: Thu 13 Apr 2023 23:59 PDT – no late submissions

Overview

Using your implementation from Coding Project #2, modify the sum_vector and sum_indirect codes as described below, run them on KNL nodes on cori.nersc.gov to collect new runtime information, and then use this new information to estimate the cache miss rate for each code at two different problem sizes.

Part 3 – Analyzing Results

Question 1. Please prepare a table with the following data (there are no charts in this project):

	# loops (integer)	Total memory bytes read (integer, divide by 1G for readability)	Total runtime (secs)	Avg Latency (nanoseconds)	Miss_rate (A % in the range 0..99.99)
Sum_vector 1M	3	0.0309	0.0297	1.0252	6.3388
Sum_vector 256M	3	0.0619	0.0582	0.9353	5.3147
Sum_indirect 1M	3	0.1238	0.1187	0.9567	5.2385
Sum_indirect 256M	3	0.2477	0.2396	0.9691	5.1954

Loops (integer) is the number of times your code called the sum() method in the >30 second testing window

Total memory bytes read: the total number of bytes your program loads from memory in the >30 second testing window

Total runtime of your program. It had better be > 30 secs. :)

Your average memory latency in *nanoseconds* computed as:
    Total elapsed runtime in seconds * 1,000,000,000 / total bytes read

Note: your average memory latency is your Average Memory Access Time (AMAT) in nanoseconds

Miss_rate: computing this number is the point of this entire exercise :)
From L18 slide 48:

(AMAT - hit_time) / miss_penalty = miss_rate

Recall that

- Hit_time is a constant (see L18 slide 55)
- Miss_penalty is a constant (see L18 slide 55)
- Both hit_time and miss_penalty are time measurements in nanoseconds

Question 2. Consider the data in your table, and answer the following questions (3 sentences max for each question):

a. Which code configuration shows the highest miss rate, and why?

The cache miss rate is affected by the memory access pattern of a program. If a program accesses memory in a contiguous pattern, the cache can store all the accessed data in its blocks, leading to a lower miss rate. However, if a program accesses memory in a non-contiguous pattern, the cache may have to evict data to make space for new data, leading to a higher miss rate.

b. Which code configuration shows the lowest miss rate, and why?

The code configuration with the lowest miss rate is sum_direct. This is because it accesses contiguous memory locations in a linear fashion, which is more cache-friendly than accessing data in a scattered fashion. As a result, the cache can fetch and store data more efficiently, resulting in a lower miss rate.

For full credit, your response needs to include reasoning about the relationship between memory access patterns and its impact on the cache miss rate.