CS 259 Lab 2

Implementation Summary -

Instruction Footprint:

func7	rs2	rs1	func3	rd	opcode
1	0	<address>	0	0	0x0B

1. **Defining the New Instruction**
   - Defined the prefetch instruction encoding using custom instruction space
   - Created a new vx_instrinsic function vx_prefetch that invokes the new instruction with an address operand

2. **Pipeline Modifications**
   - **Decode**: A new case was added to recognize the custom opcode, funct7 and funct3 values, and set the instruction type to prefetch.
   - **Execute**: Logic was added to handle the prefetch instruction, which marks the functional unit as Load/Store Unit (LSU), generate a new prefetch request without an actual load, and define the trace data for simulation purposes to track the address being prefetched.

3. **Functional Unit Changes**
   - A new boolean flag is_prefetch was defined to filter out prefetch requests from regular load and store operations.
   - Existing logic was modified to allow processing of prefetch requests without any stalls or conflicts. Prefetch requests are similar to load requests, with the difference that no value or response will be returned to the core.

4. **Cache Updates**
   - Additional logic was added to properly build the memory request being sent to the memory hierarchy, and process all the requests reaching the bank.
   - Additional parameters were added to count the prefetch request count and hit values to track the effectiveness of software prefetching.

The prefetch efficiency counter was implemented in the cache_sim.cpp as a performance metric. It was accessed at the end of each tick() function of cache simulation through logging it in the debug mode.

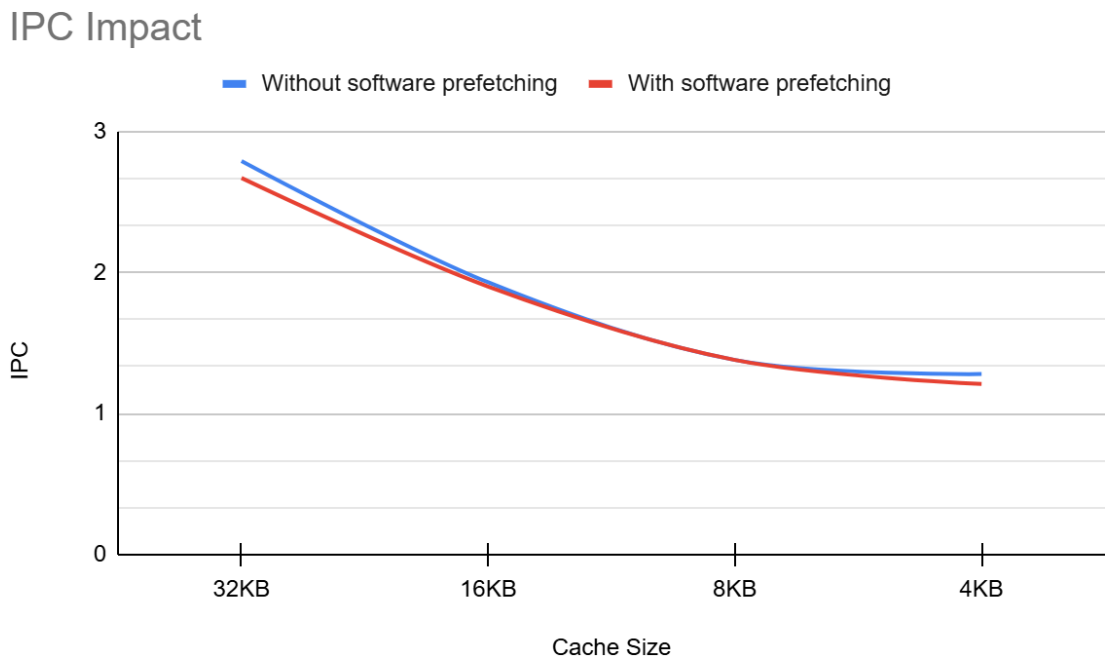
Profiling Experiments -

System Configuration:

- Number of cores = 1
- Warps x Threads = 8x8
- Number of threads = 8

Instructions per Cycle -

Cache Size	IPC (Without software prefetching)	IPC (With software prefetching)
32KB	2.79	2.67
16KB	1.93	1.90
8KB	1.38	1.38
4KB	1.28	1.21

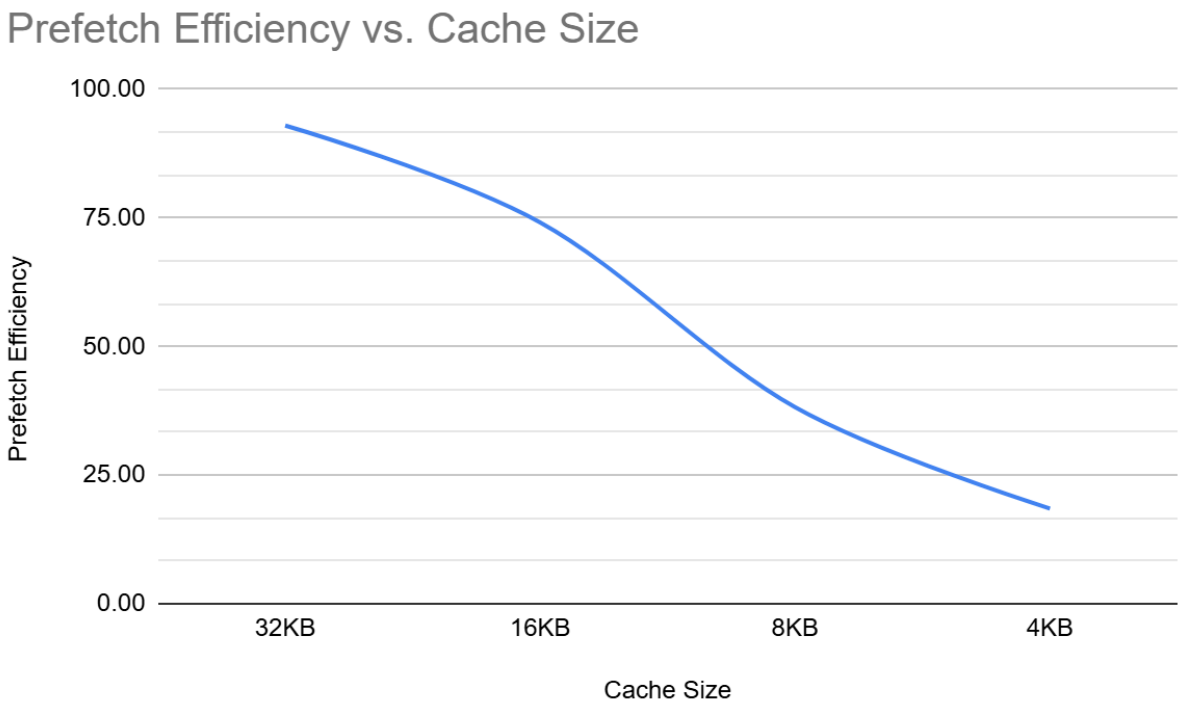


- As cache size decreases, the IPC declines due to increased cache misses. This is expected because a smaller cache can hold fewer data, increasing the probability of cache misses.

- Software prefetching does not benefit IPC in this setup. For the larger caches (32KB and 16KB), prefetching adds a slight overhead and reduces IPC. For the smaller caches (8KB and 4KB), the cache is too small to benefit from prefetching effectively, as the prefetched data might replace useful data before it can be used.

Prefetch Efficiency -
Prefetch Efficiency is defined as number of prefetch hits out of total prefetch requests sent to the memory

Cache Size	Prefetch Hits	Total Requests	Prefetch Efficiency
32KB	30000	32278	92.94
16KB	23923	32278	74.12
8KB	12332	32278	38.21
4KB	5973	32278	18.50



- Prefetch efficiency declines sharply with smaller cache sizes. A larger cache size (like 32KB) has a higher likelihood of retaining prefetched data, resulting in high efficiency. As the cache size decreases, prefetch hits reduce because the prefetched data gets evicted too soon, leading to lower efficiency.

- With smaller cache sizes, the benefits of prefetching diminish, and the overhead of prefetching may outweigh the gains. In such scenarios, prefetching strategies may need to be tuned to avoid bringing in data that has a high chance of eviction before use.