Assignment 1 ChampSim Simulator

Rahul Vigneswaran K, CS23MTECH02002

Subject: Advanced Computer Architecture (CS5363)

Reader's Notes:

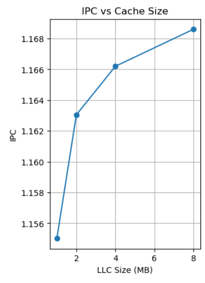
- The plots are based on average of each metric across 13 randomly chosen traces.

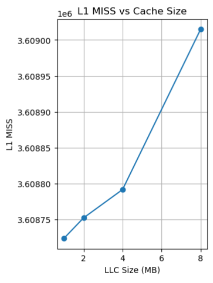
Task 1

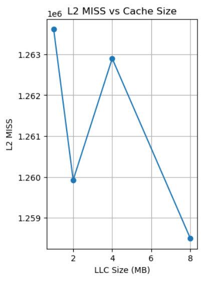
For detailed metrics collected from each of the experiments across all 13 traces and their averages, please refer to this link.

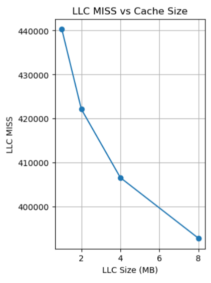
Task 2.1

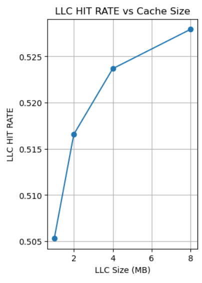
Plots

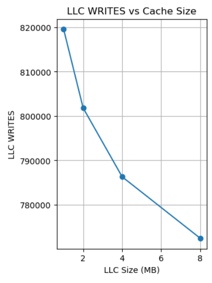


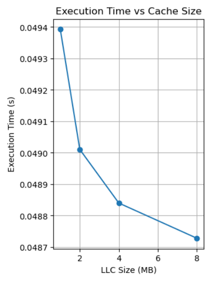












Trends & Inferences

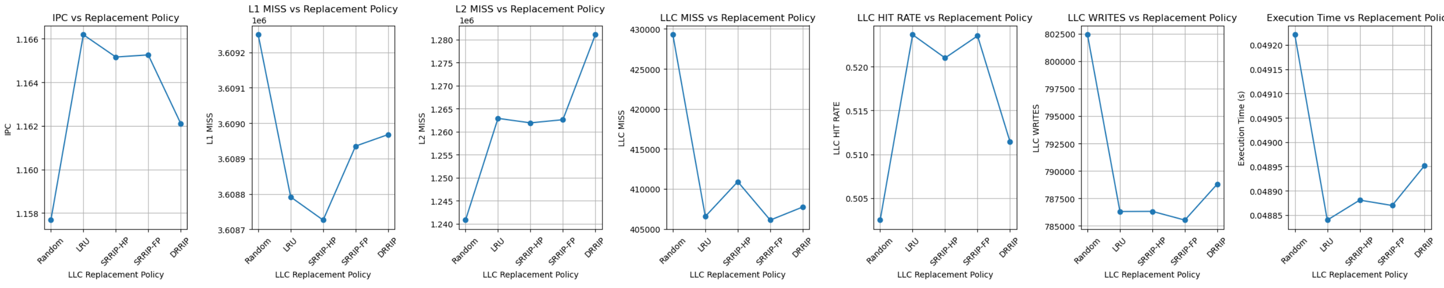
When adjusting the LLC size (1MB, 2MB, 4MB, 8MB), we observe some clear patterns:

- **IPC**: As the LLC size increases, the IPC steadily rises, indicating that a larger LLC enhances the processor's ability to execute instructions more efficiently.
- **L1 and L2 Misses**: While there appears to be some movement in the L1 and L2 miss rates, no definitive trend can be drawn from these fluctuations, likely because the changes are more sensitive to factors other than just the LLC size.
- **LLC Misses**: The number of LLC misses significantly drops as the LLC size grows, especially between 4MB and 8MB. This implies that a larger LLC can store more of the working set, leading to fewer misses at the last level of cache.
- **LLC Hit Rate**: As expected, the LLC hit rate improves with larger cache sizes, with 8MB showing the highest hit rate. This suggests that bigger caches are better at retaining frequently accessed data, boosting overall performance.
- **LLC Writes**: LLC writes decrease as the cache size increases, likely because a larger cache minimizes the need to offload data to slower memory.
- **Execution Time**: Execution time consistently decreases with larger LLC sizes, indicating that the processor spends less time retrieving data from memory, speeding up overall execution.

In summary, increasing the LLC size from 1MB to 8MB has a generally positive effect on most performance metrics. This leads to reduced cache misses, higher hit rates, and improved IPC, all of which contribute to faster execution times. The most noticeable improvements occur at 8MB, where performance metrics show significant gains across the board.

Task 2.2

Plots



Trends & Inferences

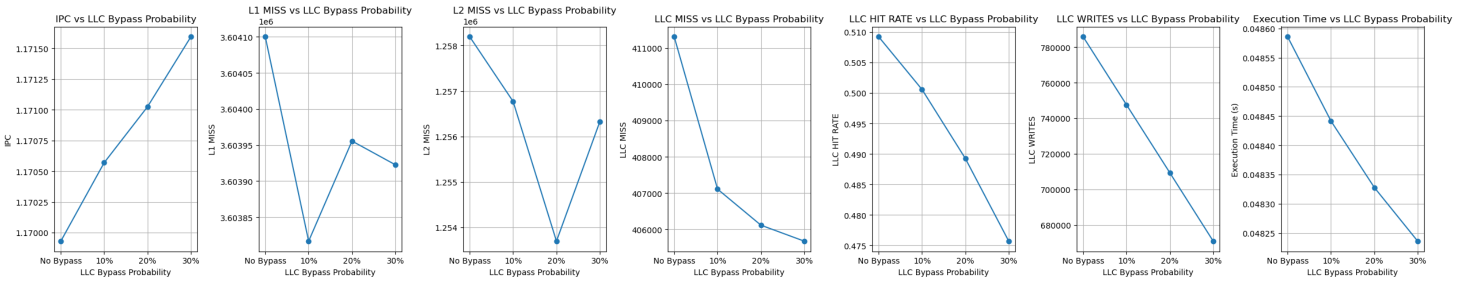
When experimenting with different replacement policies (Random, LRU, SRRIP-HP, SRRIP-FP, and DRRIP), we observe the following key trends:

- **IPC**: As anticipated, the Random replacement policy performs the worst, which is consistent with its lack of strategy in cache management. While DRRIP is known for being more adaptable across workloads, in this particular experiment, LRU outshines all other policies. This suggests that the 13 randomly chosen traces for this experiment are largely cache-friendly, allowing LRU to maximize its effectiveness.
- **L1 and L2 Misses**: The replacement policy at the LLC level doesn't appear to heavily influence L1 and L2 misses in this case. Since we are only modifying the LLC's replacement policy, and L1 and L2 caches have their own fixed policies, the variation in these metrics remains limited and inconclusive.
- **LLC Misses**: As expected, the Random replacement policy incurs the highest number of LLC misses. On the other hand, LRU performs better than DRRIP, reinforcing the earlier observation that the traces are likely cache-friendly. SRRIP-FP also shows strong performance, which suggests that some of the traces may include scanning patterns where SRRIP excels in managing cache blocks.
- **LLC Hit Rate**: Random policy has the lowest hit rate, underscoring the fact that strategic replacement policies like LRU and SRRIP lead to significantly improved cache performance. Similar to previous observations, LRU and SRRIP-HP perform well in terms of hit rates, likely due to their ability to better manage frequently accessed data.
- **LLC Writes**: LRU, SRRIP-HP, and SRRIP-FP result in fewer LLC writes compared to DRRIP, indicating that they are more efficient in managing cache data and reducing unnecessary write-backs to memory.
- **Execution Time**: LRU achieves the lowest execution time, benefiting from its efficient cache management and lower overhead compared to the other policies. Random, on the other hand, results in the longest execution time due to its ineffective caching strategy, which causes more frequent cache misses and higher latency.

When varying the replacement policy, LRU consistently performs best across the majority of metrics, achieving high IPC, low LLC misses, and a high hit rate, all of which contribute to faster execution times. SRRIP-HP and SRRIP-FP also perform well, striking a balance between fewer LLC writes and solid hit rates. On the other hand, Random show suboptimal performance, with higher LLC misses, increased LLC writes, and longer execution times. This experiment highlights that when the workload consists of cache-friendly traces, LRU is particularly effective, while policies like DRRIP may not be as well-suited.

Task 3

Plots



Trends & Inferences

Varying the LLC (Last Level Cache) bypass probabilities at 0% (No Bypass), 10%, 20%, and 30% reveals the following trends:

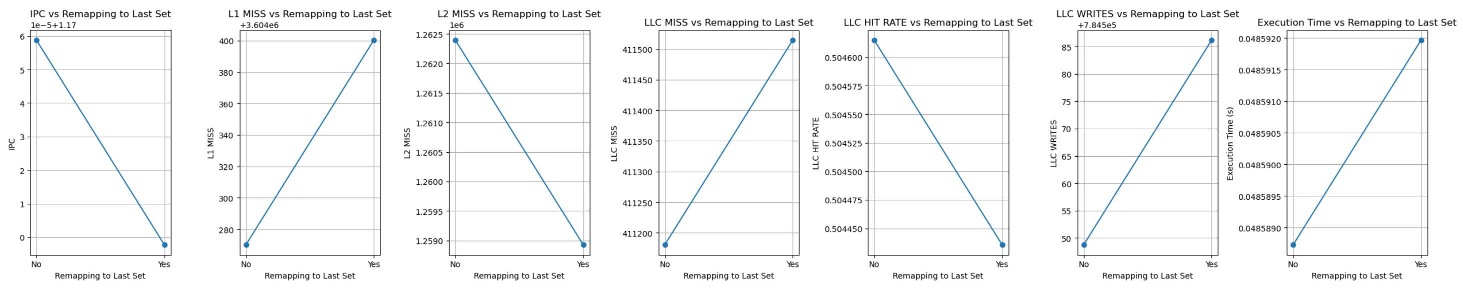
- **IPC**: As the bypass probability increases, the IPC shows a slight increase. Although the plot may suggest significant improvement, the actual value increase is minimal. It is anticipated that the IPC gains will plateau with further increases in bypass probability.
- **L1 and L2 Misses**: Bypassing affects only the LLC; therefore, changes in L1 and L2 cache misses are not particularly relevant. Any variations observed in these metrics are incidental and do not offer meaningful insights in this context.
- **LLC Misses**: Surprisingly, LLC misses decrease as the bypass probability increases. This is counterintuitive because, typically, a lower hit rate should correspond to higher miss rates. While this could be attributed to trace-specific behavior, it may warrant further investigation to understand the underlying cause.
- **LLC Hit Rate**: The LLC hit rate decreases with higher bypass probabilities. Increased bypassing reduces the chances for useful data to stay in the cache, leading to a significant drop in hit rate and overall cache efficiency.
- **LLC Writes**: LLC writes decrease with higher bypass probabilities, as fewer blocks are being written to the LLC due to bypassing.

- **Execution Time**: Interestingly, the execution time decreases marginally with higher bypass probabilities. This could be trace-specific, where bypassing aids in increasing efficiency. It is suspected that the traces involve scanning patterns, leading to blocks being evicted, thus causing an overall performance gain.

Increasing LLC bypass probabilities results in marginal improvements in IPC and execution time, possibly due to reduced cache pollution from non-reusable data. However, the decrease in LLC hit rate and cache efficiency indicates that excessive bypassing may eventually offset these gains. Further investigation is recommended to understand the unexpected decrease in LLC misses.

Task 4

Plots



Trends & Inferences

When remapping all accesses from set 0 to the last set, the following trends are observed:

- **IPC**: IPC is significantly higher without remapping. When remapping is applied, the IPC drops sharply, indicating contention or inefficiency, likely due to increased pressure on the last cache set.
- **L1 and L2 Misses**: Although remapping occurs only in the LLC, L1 misses increase and L2 misses decrease with remapping. But no clear inference can be drawn based on only this.
- **LLC Misses**: As expected LLC misses rise with remapping, likely because of increased conflict in the last set, leading to more frequent evictions and replacements.
- **LLC Hit Rate**: As expected, because of this remapping, the LLC hit rate takes a dip due to the increases in LLC miss and writes.
- **LLC Writes**: LLC writes increase significantly, suggesting that more frequent evictions are causing higher write-back activity.
- **Execution Time**: Execution time increases noticeably with remapping, indicating a performance hit, likely due to the added overhead and inefficiencies introduced by the remapping.

Remapping accesses to the last set negatively impacts performance across all metrics. The IPC drops and hit rate drops while execution time rises, LLC misses and LLC writes increase. Overall, the system performs better without remapping, as remapping introduces unnecessary overhead and contention in the last set.

Acknowledgement

I am grateful for all the invaluable discussions with Guru whose guidance helped me understand the workings of the simulator.
