CS683 Project Presentation

Hierarchy Implementation in TLB

Sanoj S Vijendra, Navnit Kumar Thakur

I use Python

22b0916@iitb.ac.in, 22b0936@iitb.ac.in

Problem statement

- To implement Translation Lookaside Buffer (TLB) designs incorporating varying hierarchical conditions, including inclusiveness, exclusiveness, and non-inclusiveness.
- To analyze and compare the performance impact of these TLB configurations on system behavior through custom-coded simulations.

Prior Works

- Limited prior research exists on implementing and analyzing Translation Lookaside Buffer (TLB) designs under varying hierarchical conditions.

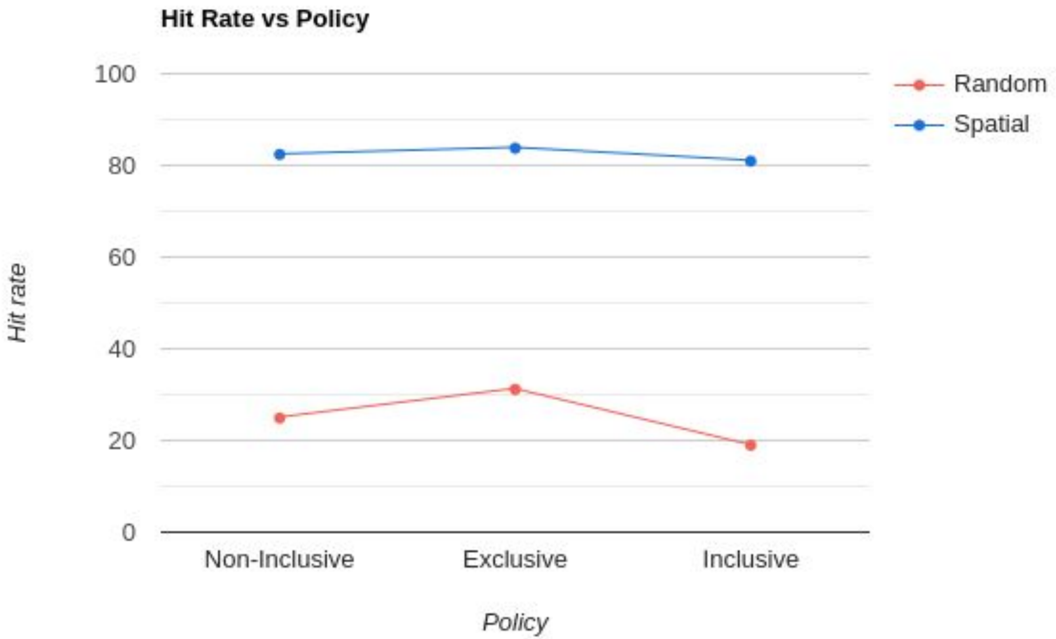
Goal of the Project

- The goal of our project is to implement hierarchical conditions in the Translation Lookaside Buffer (TLB), similar to cache hierarchy (inclusiveness, exclusiveness, and non-inclusiveness), and compare the hit rates of these configurations with a standard TLB implementation.
- After achieving this, we will implement the policy similar to Bimodal Inclusion Policy for exclusiveness, inclusiveness, and non-inclusiveness, and evaluate the hit rates for each configuration.

Work done so far

- Due to challenges in implementing the TLB in the ChampSim simulator, we have opted for an alternative approach. We developed a basic implementation by modifying and extending the existing TLB code available in a GitHub repository. This allowed us to create a working prototype while overcoming limitations in directly integrating with the simulator.
- The code provided a basic naive TLB implementation along with a randomly generated dataset, which was used to calculate the hit rate for evaluating the TLB performance.
- We modified the code to implement inclusiveness, exclusiveness, and non-inclusiveness in the TLB, along with initial efforts to incorporate the policy similar to BIP.

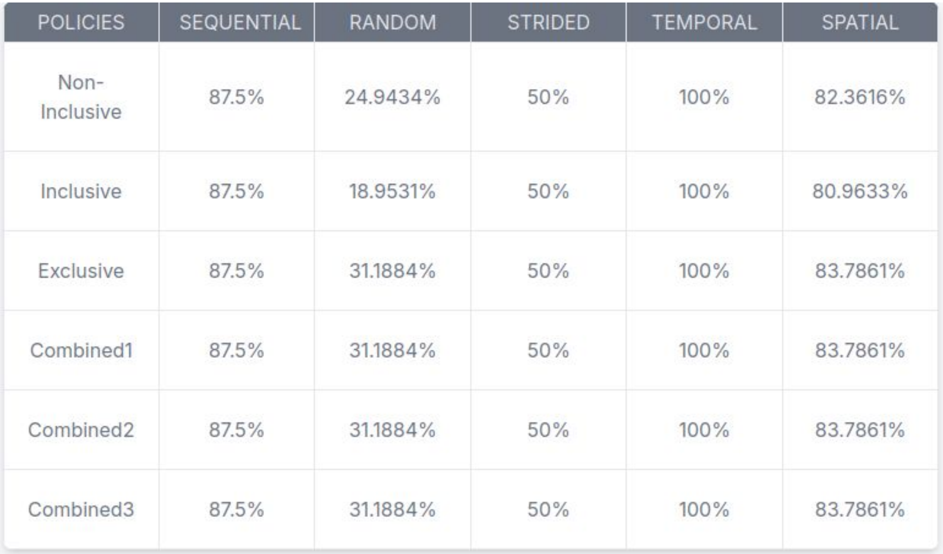
Results



- In general, exclusive policy works best.

Results

Table (Hit Rate for Different Policies and WorkLoads)



POLICIES	SEQUENTIAL	RANDOM	STRIDED	TEMPORAL	SPATIAL
Non-Inclusive	87.5%	24.9434%	50%	100%	82.3616%
Inclusive	87.5%	18.9531%	50%	100%	80.9633%
Exclusive	87.5%	31.1884%	50%	100%	83.7861%
Combined1	87.5%	31.1884%	50%	100%	83.7861%
Combined2	87.5%	31.1884%	50%	100%	83.7861%
Combined3	87.5%	31.1884%	50%	100%	83.7861%

- Combined1 = 0.2(non-Inclusive)+0.2(Inclusive)+ 0.6(exclusive)
- Combined2 = 0.33 each
- Combined3 = 0.1(non-inclusive) + 0.2 (inclusive)+0.7 (exclusive)

Results

- For sequential, strided (constant) and temporal locality workloads, the hit rates for all the policies were found to be same.
- Combined policy behaves almost same for every workload with non-zero exclusive component.
- When the exclusive component is zero, it behaves as inclusive unless the inclusive component is also zero.

Github link

- Project github repo can be found here.