GPGPU L1D Cache Analysis

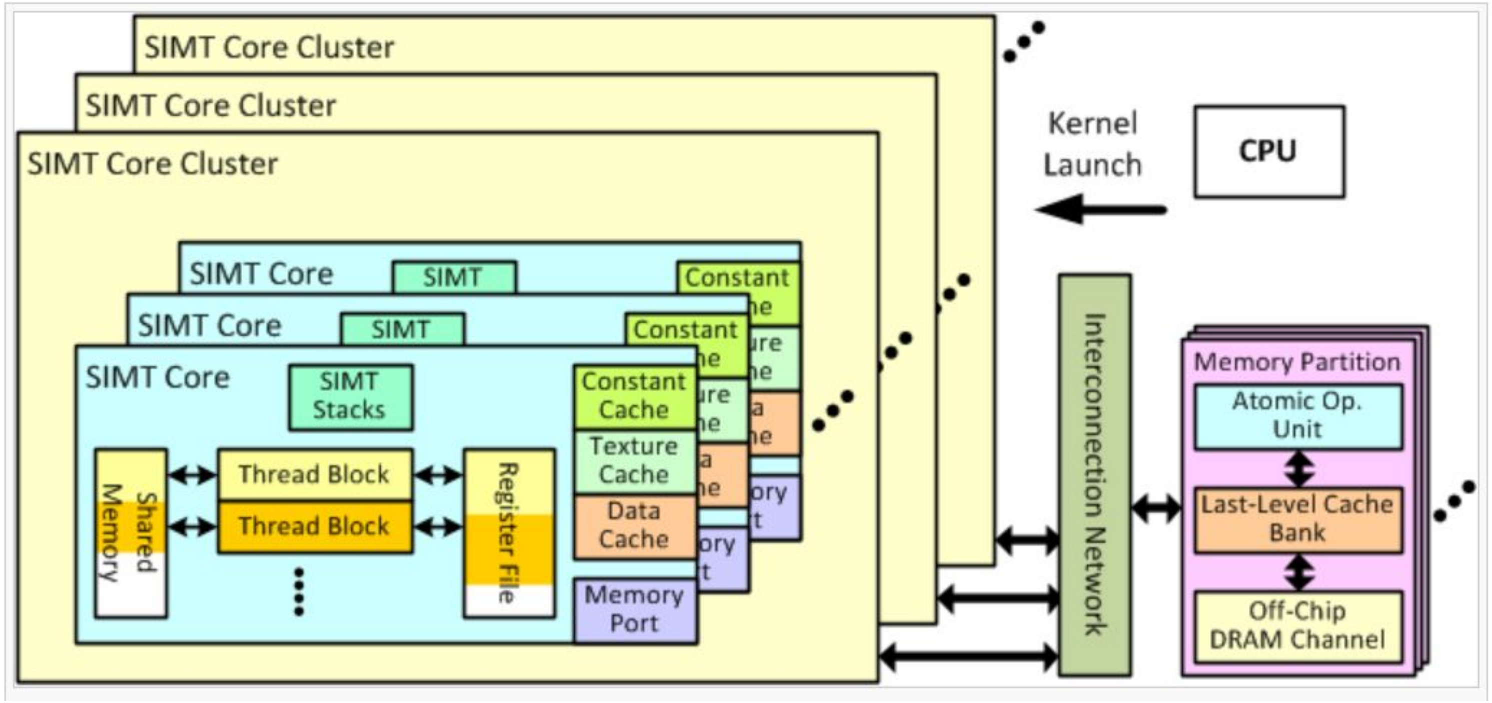
CSE 530 Class Research Project

Presented by:
Pranitha Malae
Rishabh Jain

Motivation

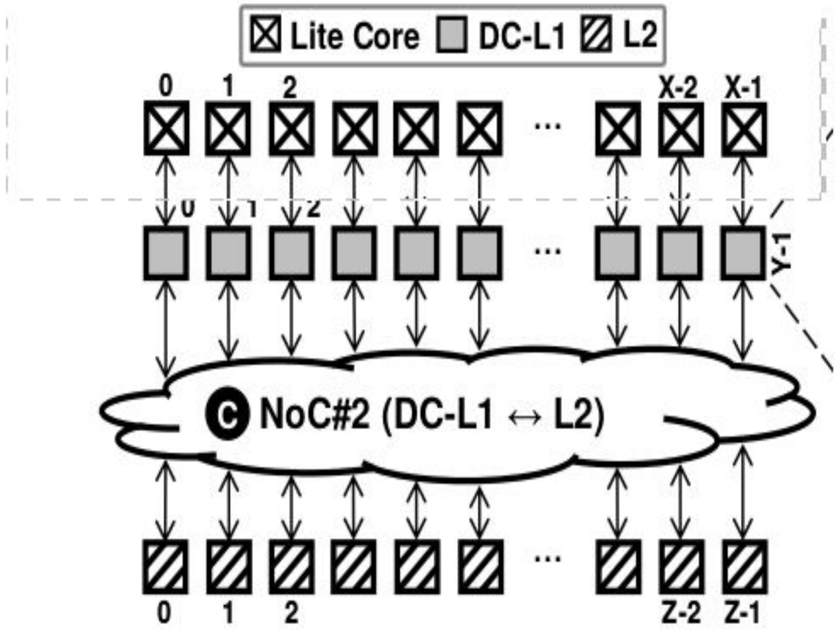
- Many general purpose applications are running on GPGPUs
  - Data or image processing, ML training etc.
- Memory hierarchy is important for overall performance.
- Measurement of following parameters
  - IPC
  - L1D Cache miss rate
  - Replication ratio
  - NoC traversal latency
- GPGPU-sim

GPGPU-sim Architecture

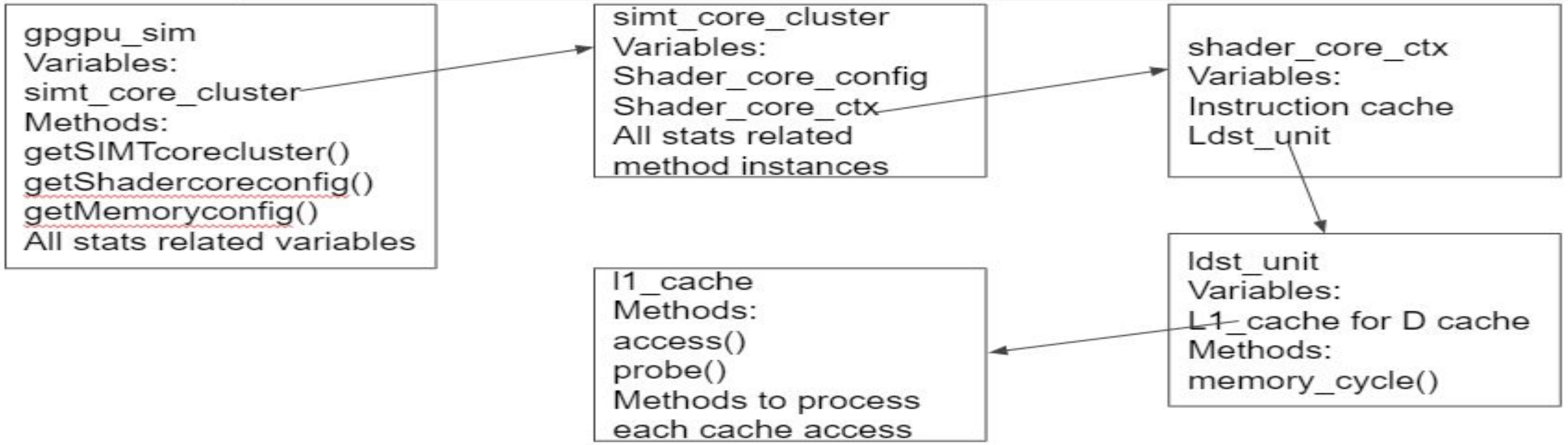


Replication observation:

Replication Ratio: The ratio of L1 misses that can be found in other L1 caches to total L1 misses



Findings in GPGPU-sim code



- Both L1 and L2 caches uses same method for cache access.
- Cache misses are handled using mshr (miss status holding registers) tables.
- To record statistics separate methods are present.
- Configurations are present in gpgpusim.config and config_fermi_islip.icnt.

Modifications in the code

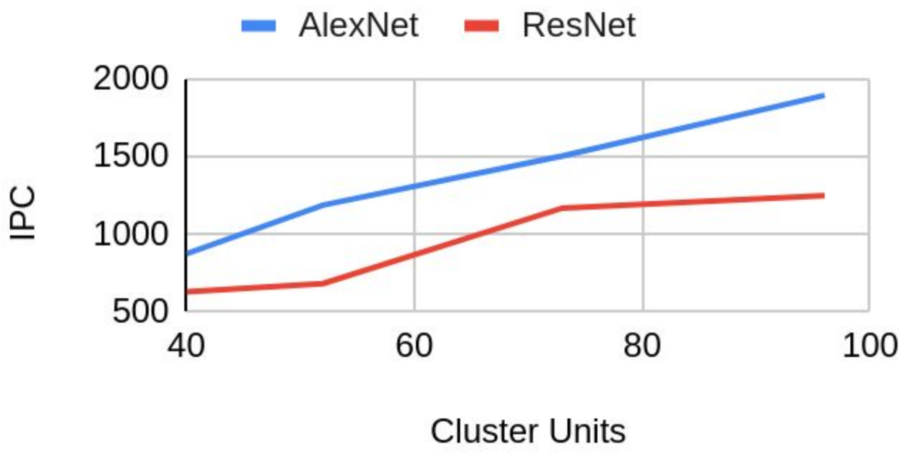
- Main idea is to check whether a missed L1D cache line of one core is present in other core L1D cache or not.
- There is a common global parameter called ‘gpu’ for all clusters
- Using this object we obtained L1D cache object of all clusters
- Using this L1D object we **probed** the cache.
- Example code for probing:

    -
      ```
      l1_cache* temp =
      this->m core->get_gpu()->getSIMTCluster()[curr_cid]->get_shader_core_object()[0]->m_ldst_un
      it->m L1D
      ```

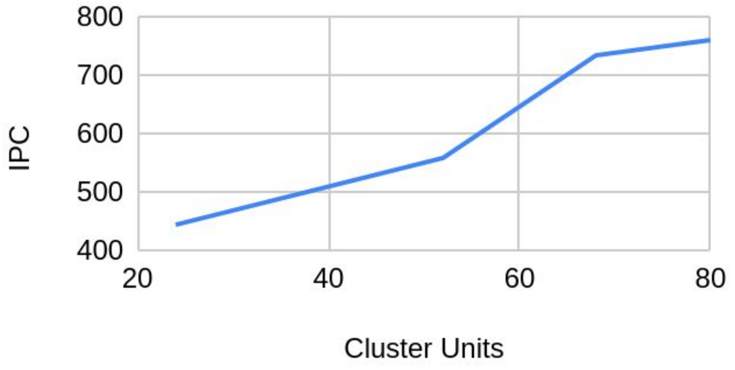
    -
      ```
      temp_status = temp->m_tag_array->probe(blk_addr,cache_idx);
      ```

Role of Cluster Units on IPC

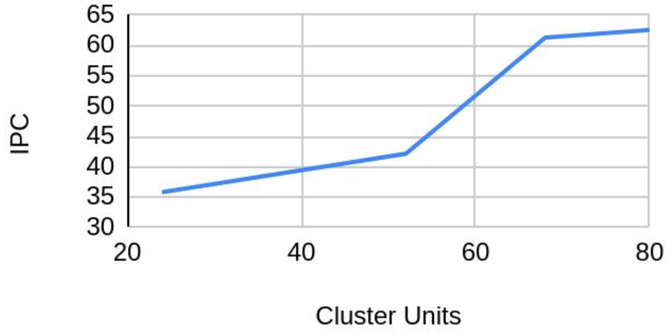
IPC vs CUs



RAY

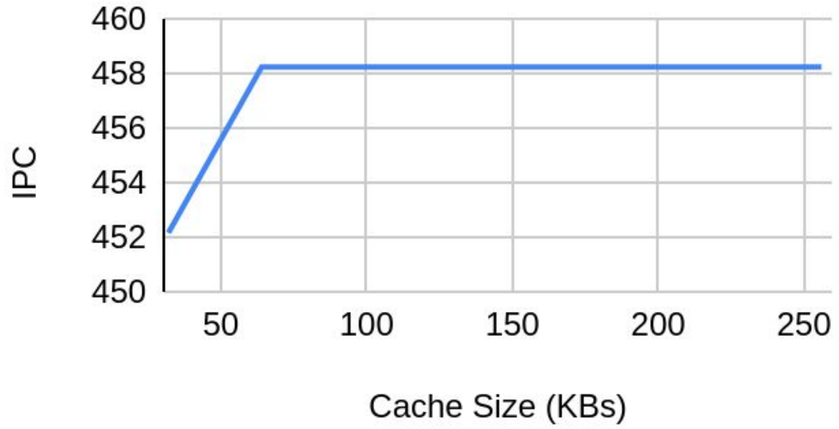


BFS

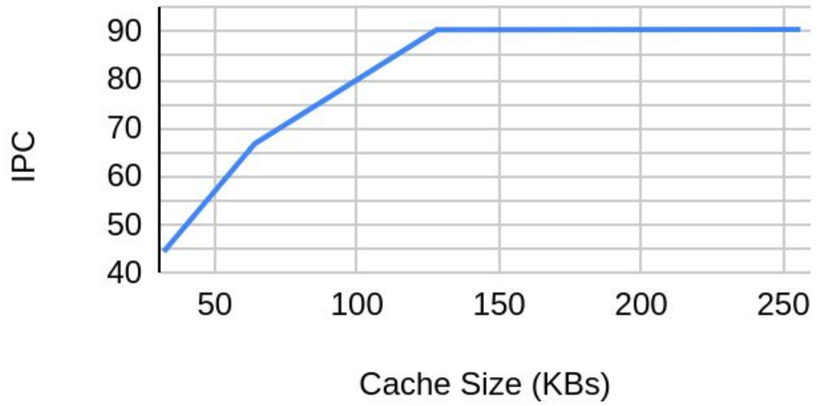


Role of Cache-Size on IPC

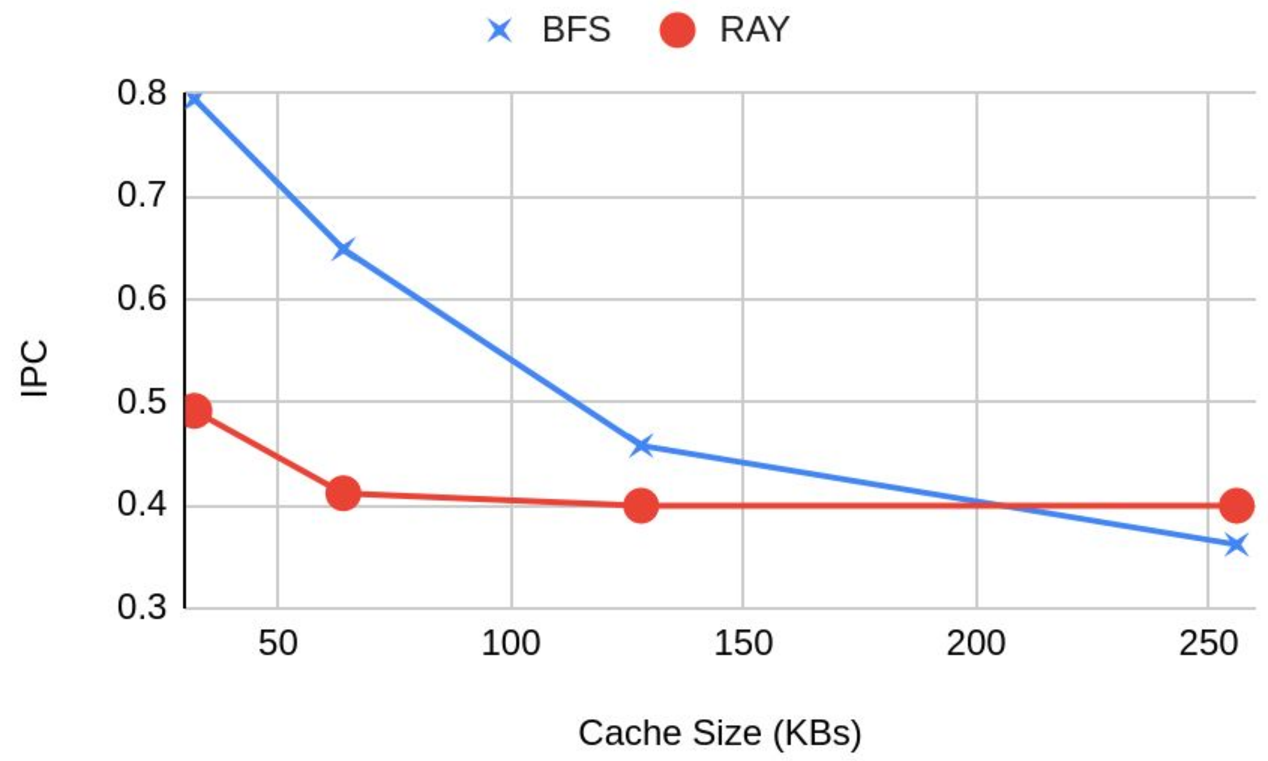
RAY



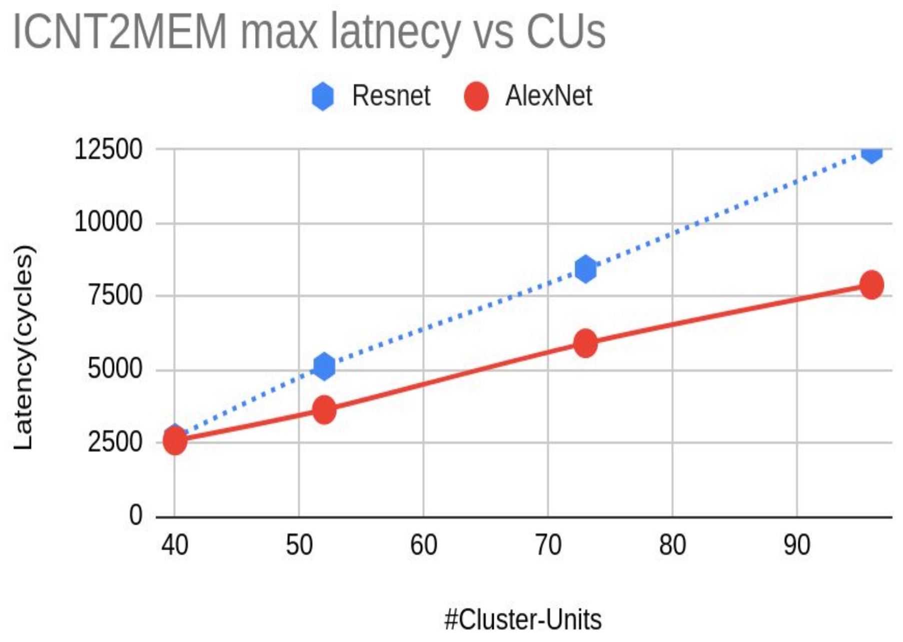
BFS

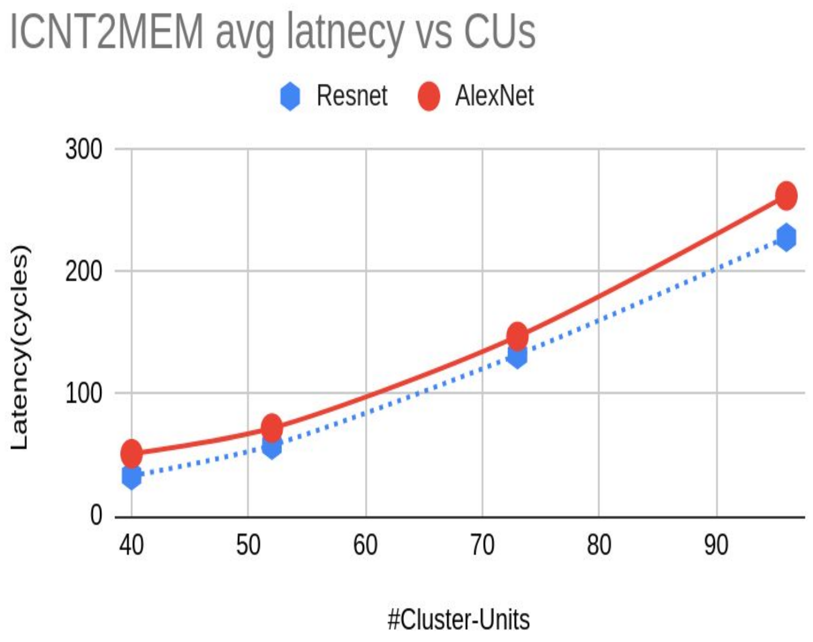


Role of Cache-Size on Cache-miss-rate

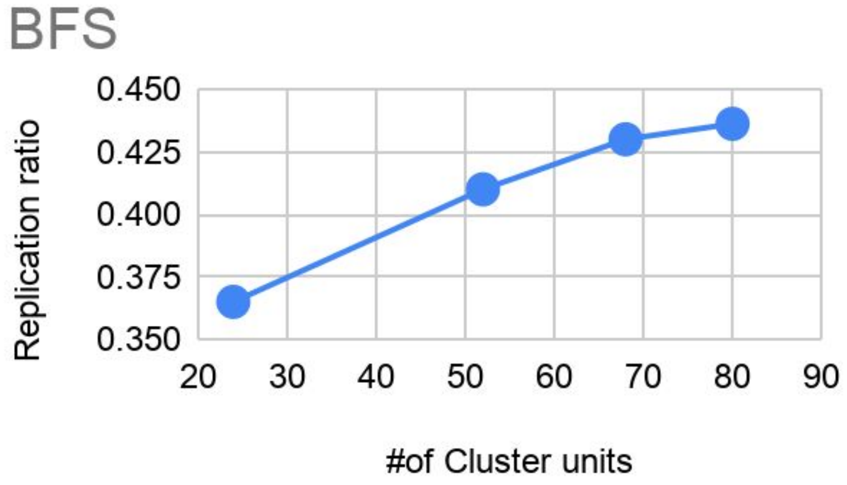


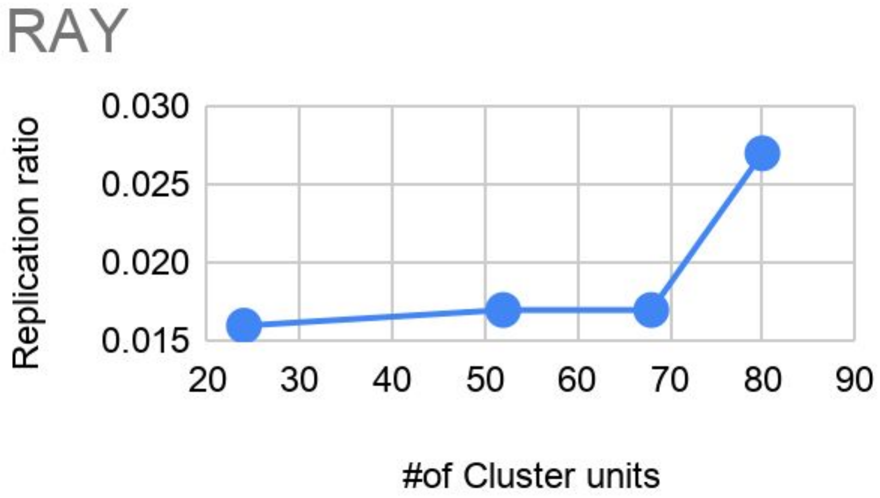
Role of Cluster Units on NoC



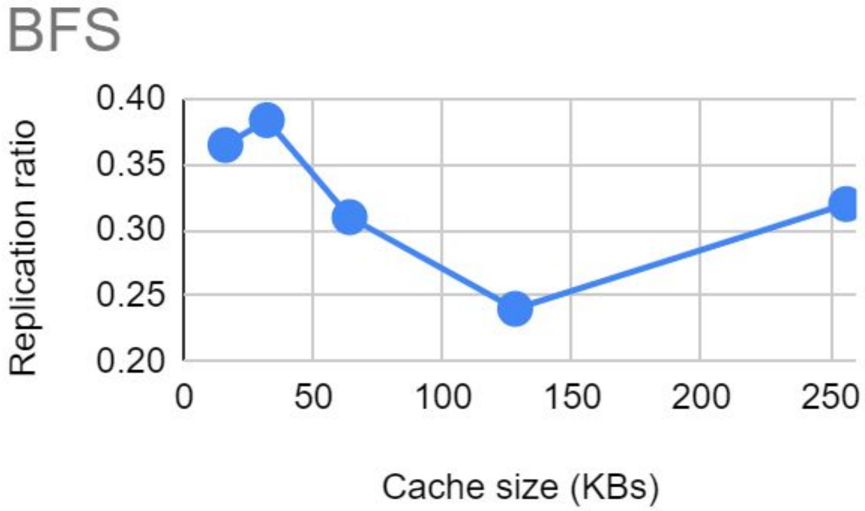


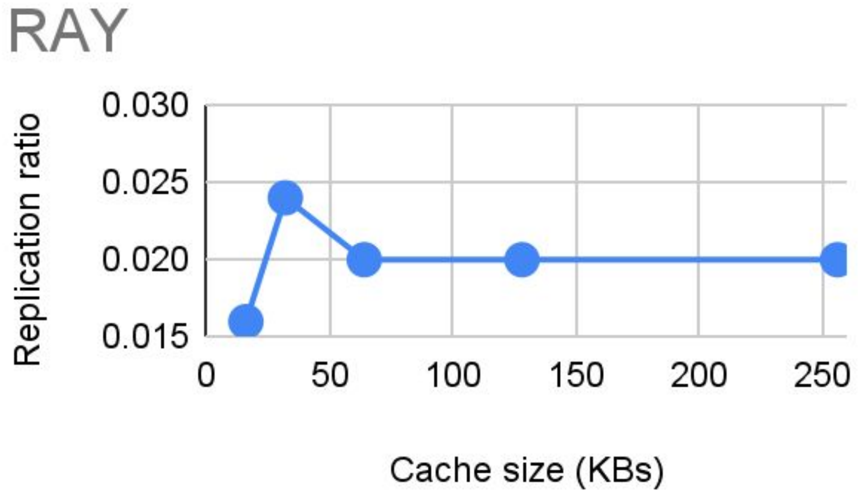
Role of Cluster units on Replication ratio





Role of Cache-size on Replication ratio





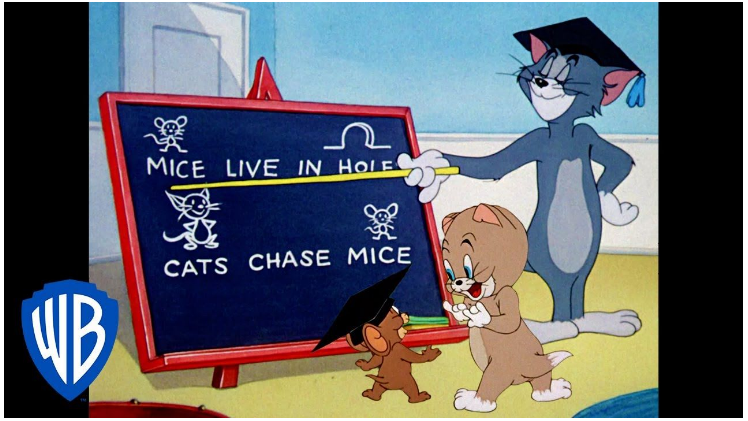
Other observations:

- Nearly uniform distribution of insts over multiple cores
- Nearly uniform distribution of cache access over multiple cores
- Some benchmarks like LSTM, GRU(gated neural network) used only 1 and 2 cores.
- Didn’t observe a concrete trend for #ClusterUnits vs cache-miss-rate

Mitigating replication and Future work:

- Workload characterization with more benchmarks, new metrics and new performance counters to obtain new insights.
- Improving the reference paper: Coming up with the right decoupled cache grouping schemes for various applications which could motivate for a dynamic cluster design.
- HW-SW codesign: understanding program behavior causing replication → using compiler to mark these addresses → designing hardware which can use these markers to smartly store data in private cache.
- Literature study of management of shared vs private components in existing microarchitecture works.
- Due to high network latency for high CUs, coming up with better NoC could help in performance.

Lessons learnt



- Figuring out metrics for sensitivity studies
- Installation of a simulator can be challenging.
- Playing with GPGPU-sim configurations and grepping stats
- Hacking codebase to put our replication counters
- Running simulations. Fun fact: //ly running the time consuming bmarks(> 1 day)
- Drawing graphs - connecting observations with architecture concepts
- Able to replicate a couple of results shown in reference paper.

THANK YOU

Please ask questions if any