7. CONCLUSION AND FUTURE WORK

7.1. Conclusion

When it comes to concurrent multi-threaded applications in a CMP environment, LLC plays a crucial role in the overall system efficiency. Conventional replacement strategies like LRU, NRU etc may not be effective in many instances as they do not adapt dynamically to the data requirements. Numerous researches are being conducted to find novel ways for improving the efficiency of the replacement algorithms applied at LLC. In this dissertation, three novel counter based replacement strategies have been proposed.

- A novel replacement algorithm called CB-DPET has been proposed which takes a more systematic and access-history based decision compared to LRU. It is further split into DPET and DPET-V. DPET associates a counter with every cache block to keep track of its recent access information. Based on this counter’s values, the algorithm makes replacement, insertion and promotion decisions.

- A dynamic and a structured replacement technique called LCP has been proposed to be adopted across the LLC. Here the elements of the cache are logically partitioned into four zones based on their likeliness to be referenced by the processor with the help of a 3-bit LCP counter associated with every cache line. Replacement candidates are chosen from the zones in the increasing order of their likeliness factor starting from the NLR zone. For every hit, the corresponding element is moved up by one zone by adjusting the LCP counter value. If it has reached the top most zone (MLR) the counter value is left untouched. For every miss the counter value is decremented by ‘1’ to prevent stale data items from polluting the cache.

- A novel counter based prioritizing algorithm called SHP has been proposed to be applied across the LLC. Here every cache block is associated with a 2-bit sharing degree counter which iterates from 0 to 3. A dynamic software based TT filter is associated with every block to keep track of the threads that are
accessing that block. A hit counter is used to keep track of the hits received by the data item. Values of the sharing degree and the hit counters are used to compute a priority for each cache block. This priority is then used to make judicious replacement decisions.

<table>
<thead>
<tr>
<th>Method</th>
<th>2-Core</th>
<th>4-Core</th>
<th>8-Core</th>
</tr>
</thead>
<tbody>
<tr>
<td>CB-DPET</td>
<td>8%</td>
<td>4%</td>
<td>3%</td>
</tr>
<tr>
<td>LCP</td>
<td>6%</td>
<td>9%</td>
<td>5%</td>
</tr>
<tr>
<td>SHP</td>
<td>7%</td>
<td>9%</td>
<td>5%</td>
</tr>
</tbody>
</table>

Table 7.1 provides the percentage improvement obtained in overall number of hits when compared to LRU for all the three methods. It is observed that up to 9% improvement has been obtained at the L2 cache level (for LCP) in overall hit percentage increase and an IPC speed up of up to 1.08 times that of LRU (for SHP).

Simulation studies shows that all the three proposed counter based replacement techniques have outperformed LRU with minimal storage overhead.

7.2. Scope for Further work

Dynamic replacement strategies have proven to work well with versatile set of application workloads that currently exist. Increasing the dynamicity without inducing much hardware overhead is crucial. This research has experimented novel replacement strategies on the shared L2 cache while the modifications that may be required to run the same algorithms in the relatively smaller private L1 cache is an interesting area to explore.

Appending additional bits per cache block in CB-DPET can help in storing more information about the access pattern of the data item but storage overhead needs
to be taken into consideration. In LCP, optimization of the victim search process could be done to expedite the replacement candidate selection. Multi-threading could be a better solution here. Instead of having one thread searching the cache set in multiple passes there can be multiple threads searching in different zones but inter-thread communication may have to be explored to make sure the process stops once victim is found. In sharing and hit-based prioritizing algorithm, replacement priority of cache blocks is computed from the sharing degree and number of hits received. In order to boost the accuracy, various other parameters like miss penalty incurred while replacing that block etc can also be considered while computing priority.