CS6600: Computer Architecture
Assignment 2: Design your own LLC Cache Replacement Policy

Bala Dhinesh - EE19B011
Vishnu Varma V - EE19B059

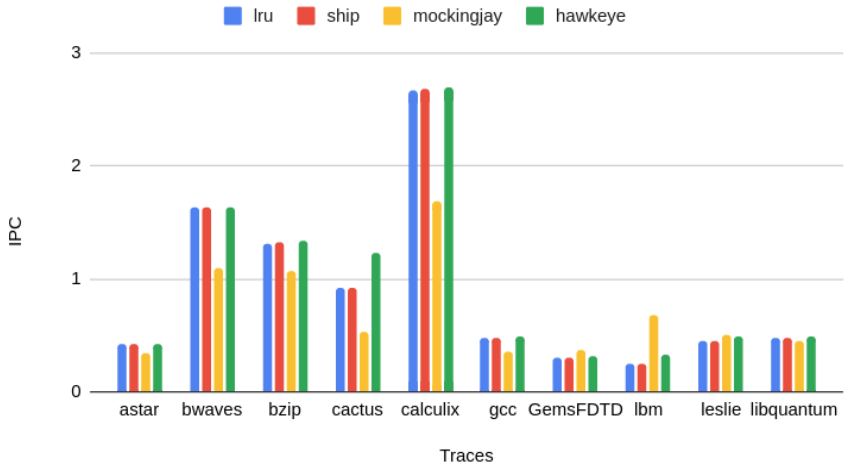
Introduction:

The assignment aims to design our cache replacement policy and benchmark it against current state-of-the-art policies like LRU, SHiP, Hawkeye, and Mockingjay. Many methodologies exist for cache replacement as trivial as a random replacement, queue-based, recency-based, frequency-based, and RRIP policies, Belady's optimal algorithm, up to as extensive as machine learning. Some replacement policies based on Belady's algorithm are studied and evaluated along with other well-known replacement policies below.

Brief description of the state-of-the-art policies:

The algorithms implemented like hawkeye and Mockingjay are based on Belady's optimal algorithm and improvements upon it. Belady's algorithm requires knowledge of the future, it evicts the cache lines that will be reused farthest in the future, which makes it infeasible in general, but hawkeye implements this algorithm by predicting the future based on experiences from past accesses. The data, whether the access would be cache friendly or cache averse, is fed into RRIP, which provides a corresponding RRPV value based on which the cache lines are selected for eviction. Mockingjay, on similar lines, improves upon hawkeye by making more granulated decisions, and the decision of eviction would await until more information is available. The decisions are based on the ETR (Estimated Time of Reuse) of cache lines. Using these methods, results close to the optimal Belady's algorithm could be obtained.

Performance of various LLC Replacement policies:



Analysis:

For the given traces, the first six traces follow a pattern where Mockingjay is incomparable to the rest of the policies (exception includes that for cactus, hawkeye outperforms other policies), while the other four replacement policies are very close, which is not precisely distinguishable. However, Mockingjay is comparable to other policies for the remaining four traces and exceeds others for LBM trace. Hawkeye performs best in comparison to the policies that are taken under consideration, as it can be seen that for the first six traces, hawkeye provides either the best IPC or stays one among the best, following for the remaining four traces, it is one among the best or second best after Mockingjay. Based on theoretical evidence, Mockingjay should perform better, but it is not universal, and it is particular to the traces under consideration; so for the given traces, there is no predetermined trend for the replacement policies rather, it varies with traces considered and the variation can be seen in the above plot.

For the given traces, if we have to conclude on the performance of the already existing replacement policies, then the average IPC among the replacement policies can be considered to measure the efficacy of the policies. Taking into account the average IPC value for all traces, it can be noticed that Hawkeye performs the best, followed by SHIP, LRU, and Mockingjay. Note that this trend is particular to the given benchmark traces specific to the set of applications they are intended for.

Our replacement policy:

From the IPC values obtained for the given ten traces, we can notice that the performance of LRU is marginally lower in performance than Hawkeye. We know that LRU works best only when there is cache data access which is recency based. Since LRU and Hawkeye provide very similar results, we can infer that the prediction done by Hawkeye policy closely aligns with the recency-based policy(LRU). So our LLC replacement policy uses the best of both worlds - LRU and Hawkeye by integrating them. We declare two variables in our algorithm, *lru_counter,* and *hawkeye_counter.* Whenever we notice a cache hit in a particular policy, we increment its corresponding counter value by one and decrement if it's a cache miss.

```
if(policy){
   // If LRU policy is enabled
   if(hit){
      lru_counter++;
   }else{
      lru_counter--;
   }
}
else{
   // If Hawkeye policy is enabled
   if(hit){
      hawkeye_counter++;
   }else{
      Hawkeye_counter--;
   }
}
```

We can decide the policy based on its counter values. If the counter value of LRU is higher, we enable the LRU policy, else Hawkeye policy. This ensures the best policy usage, which is prone to better results at a particular time.

```
if(lru_counter > hawkeye_counter){
   policy = true;      // enables LRU
}else{
   policy = false;     // enables Hawkeye
}
```

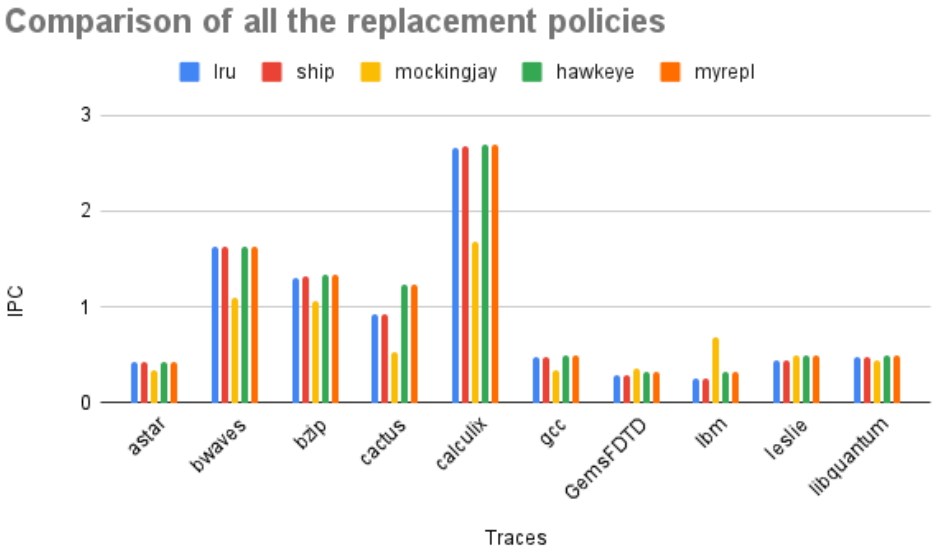
Advantages:

- This policy learns LRU and Hawkeye policy and enables them accordingly.
- This results in a slight improvement in the performance over Hawkeye.
- It is adaptive to the nature of accesses and chooses the best policy to provide better performance.
- It can be extrapolated to obtain a more sophisticated replacement policy that takes advantage of the different basis on which various state-of-the-art replacement policies are built.

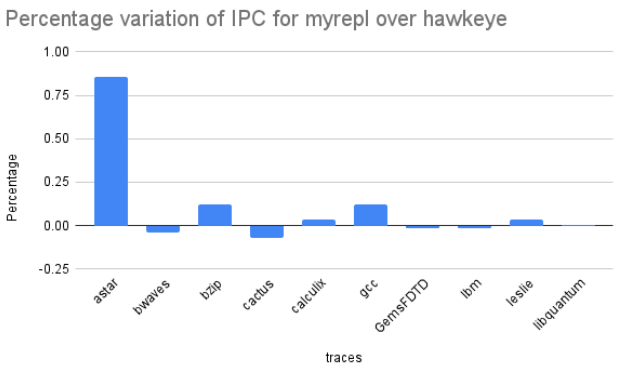
Limitations:

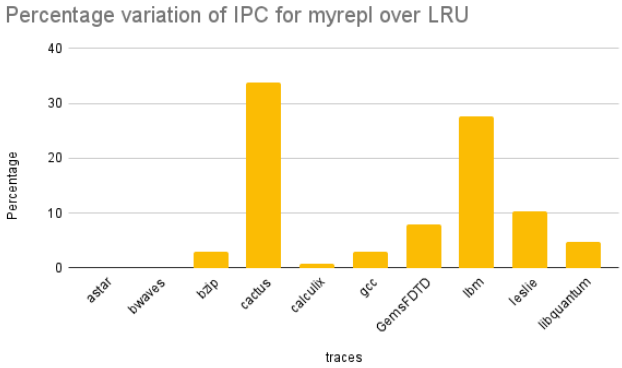
- The analysis is based on the performance obtained only from the ten given traces. If we try for a different trace that performs poorly using LRU(not recency based), this policy might degrade performance by a small amount.
- This slight degradation in the performance can be overcome by introducing a higher weight increment in the *hawkeye_counter* value(increment *hawkeye_counter* by a value greater than one).

Comparison of our replacement policy with existing replacement policies:



Comparing our replacement policy with the best among already existing policies for the given traces i.e Hawkeye. We can notice that for most of the traces (except 3 of them) there is marginal improvement in the IPC, even for the 3 cases where it decreased, the decrement is not significant. So this replacement policy is slightly better than the best policy for the given traces. Analysis of Hawkeye being the best policy for given traces has been carried out earlier. So determining that our replacement policy performs better than hawkeye implicates that it is better than other existing policies as well for the given traces. The reason for our replacement policy to be marginally improved is that, since these traces support applications where hawkeye policy is predominantly efficient, our replacement policy considers the policy that is more efficient for the given scenario, so based on it for most of the parts the predictor would predict to use Hawkeye instead of LRU, which supports the reason for the IPC to be close to that of hawkeye. If an unbiased example is considered then the improvement in the replacement policy can be determined more profoundly, as it would take advantage of both the replacement policies equally effectively.





A comparison of our replacement policy to hawkeye is already stated above to describe it as an improvement to the best policy, which in turn is the best policy in hand now while comparing it with the LRU policy provides even further insights as to how the policy would perform under various constraints.

It can be noticed that there is a significant improvement in our replacement policy when compared to LRU, this impact can be attributed to the efficacy of the traces towards hawkeye policy than that of LRU policy, so the performance of our replacement policy which comprises of both of them when compared to their individual counterparts tend to have a slight improvement over hawkeye (which is already efficient for given traces) while a significant improvement over LRU (which is comparatively not very efficient for the given traces).

Conclusion:

- Various methodologies for cache replacement policies were studied.
- State-of-the-art policies like Hawkeye and Mockingjay were studied in detail and their performance was compared with other replacement policies like Ship and LRU.
- The motive of our replacement policy was detailed, explained the advantages and limitations of our replacement policy.
- Our replacement policy was compared with hawkeye which was deemed to be better for the given traces and analyzed the performance of our replacement policy over other existing replacement policies.