WRITE COUNT PREDICTION FOR ENCRYPTED MAIN MEMORY

Team 3

Team Members

- Ajit Mathew
- Daulet
- Sudha Ravali
- Pranavi
- Xiaolong

Objective

To create a memory system to predict the write count of a memory location before the location is actually fetched into the main memory.

Task

- **Implement a lossy hash table to model Last Level Cache in C++**
- **Implement data structures to store write count history, predicted write count**
- Generate address traces using benchmarks
- **Implement Prediction Algorithm based on the history**
- **Vary parameters to find optimal prediction coverage.**

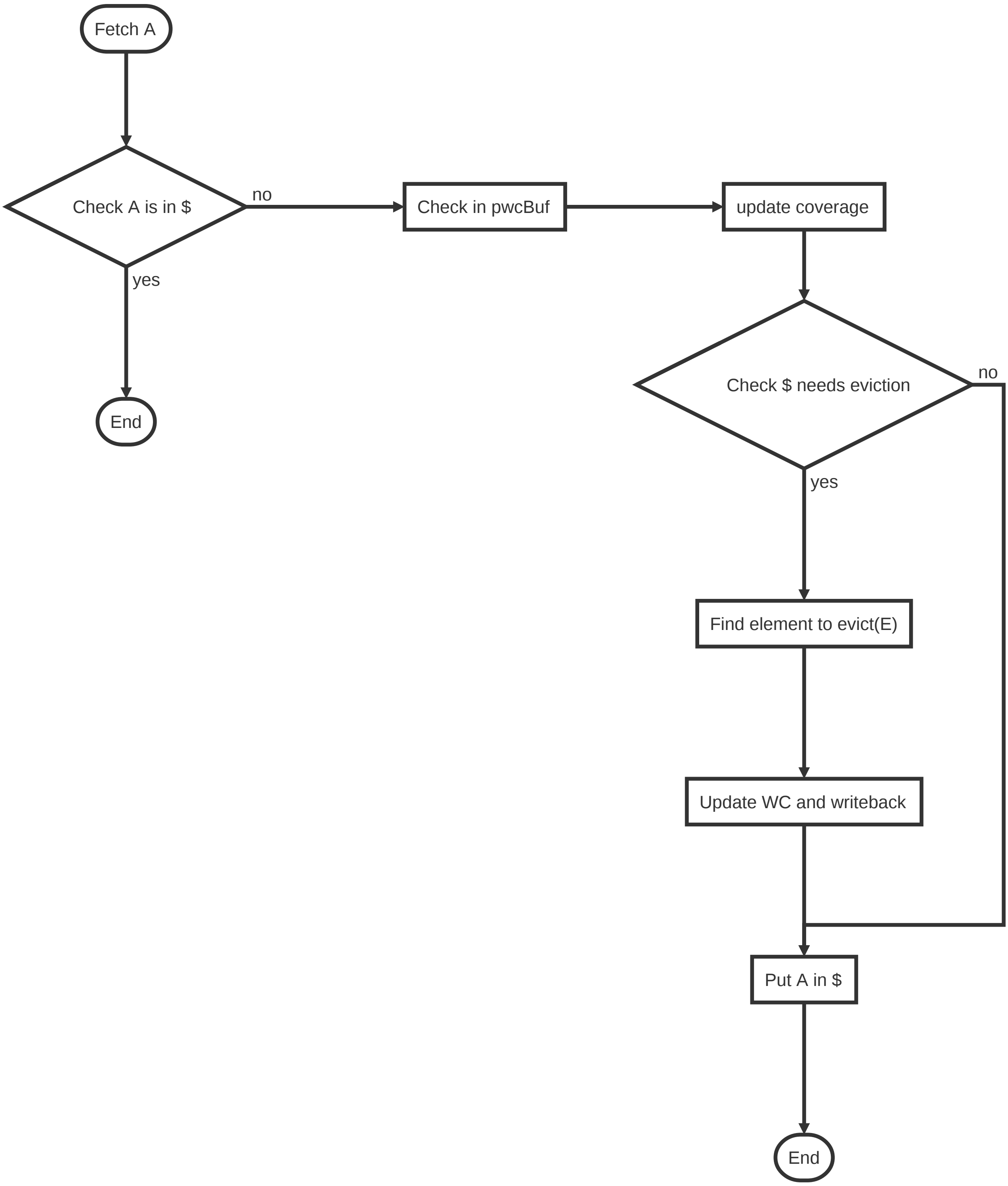
Note: Tasks in bold has been completed successfully

Design

The design is divided into three main component:

- **Main Memory**
- **Cache with Write History Holder**
- **Predicted Write Count Buffer** (pwcBuf): This buffer stores the predicted write count. On a cache miss, the buffer is read and compared to actual write count
- **Pattern Fifo**: This is used to keep track of cache miss history.

Prediction Algorithm



```
def putinCache(A):
    patternFifo.size() == HISTORY_SIZE:
        x = patternFifo.pop()
        UpdateWCHistory(x)
    patternFifo.push(A)
```

Parameters

Our model is completely parameterized. The parameters we chose to analyse our design are:

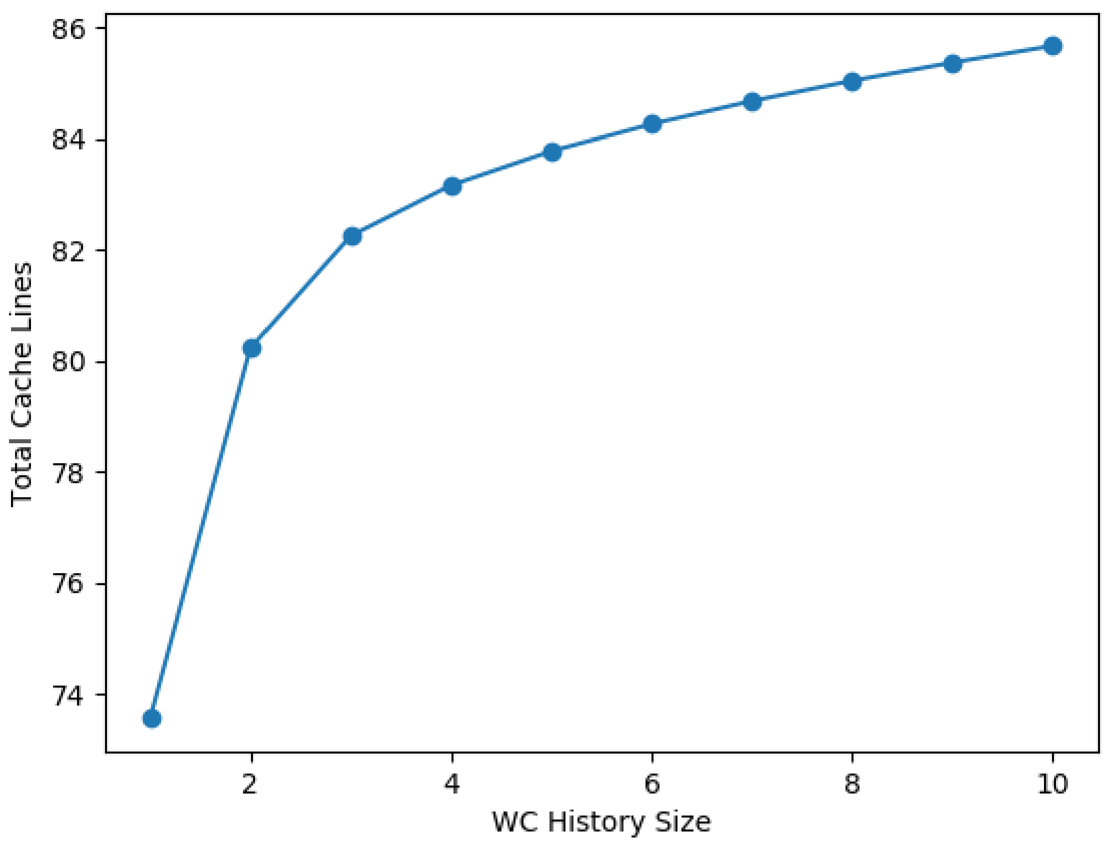
- **Write Count History Size**: This determines the size of *write count history holder (WCHH)*.
- **Range Size**: This determines the range of predicted write count history. For example if write count of **B** according to **WCHH** of **A** is n, the n to n+r values will be used to predict the write count of B.
- **Set Size**: This determines the number of cache line that can be stored in on cache set.

Results

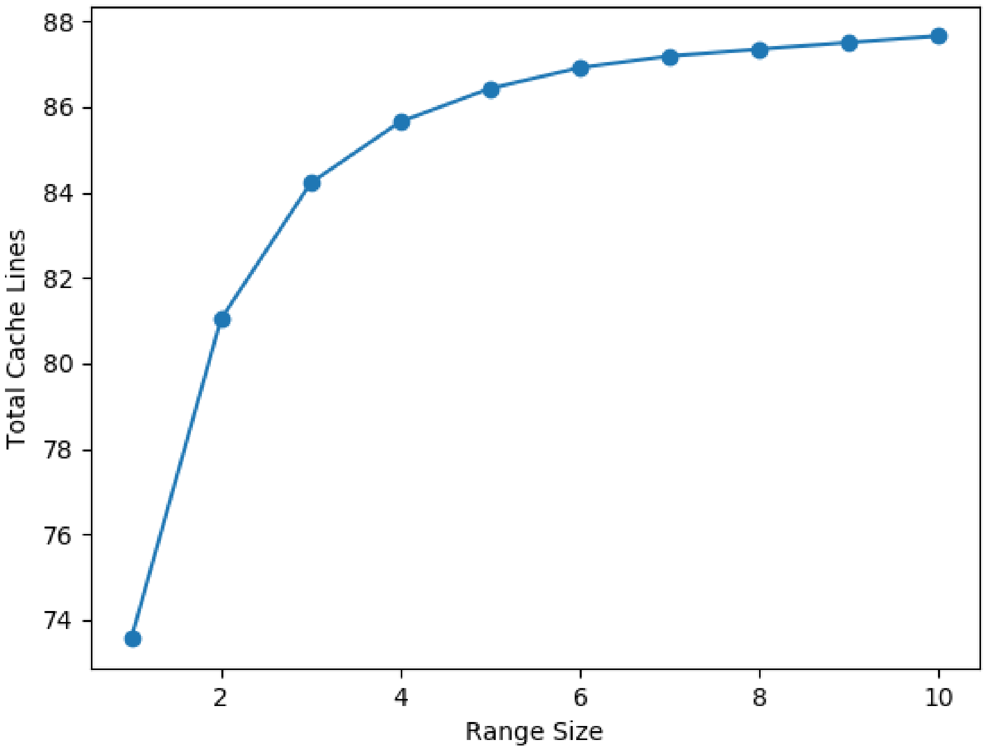
We ran our model on **NAS Parallel Benchmark** for three different workloads.

Conjugate Gradient

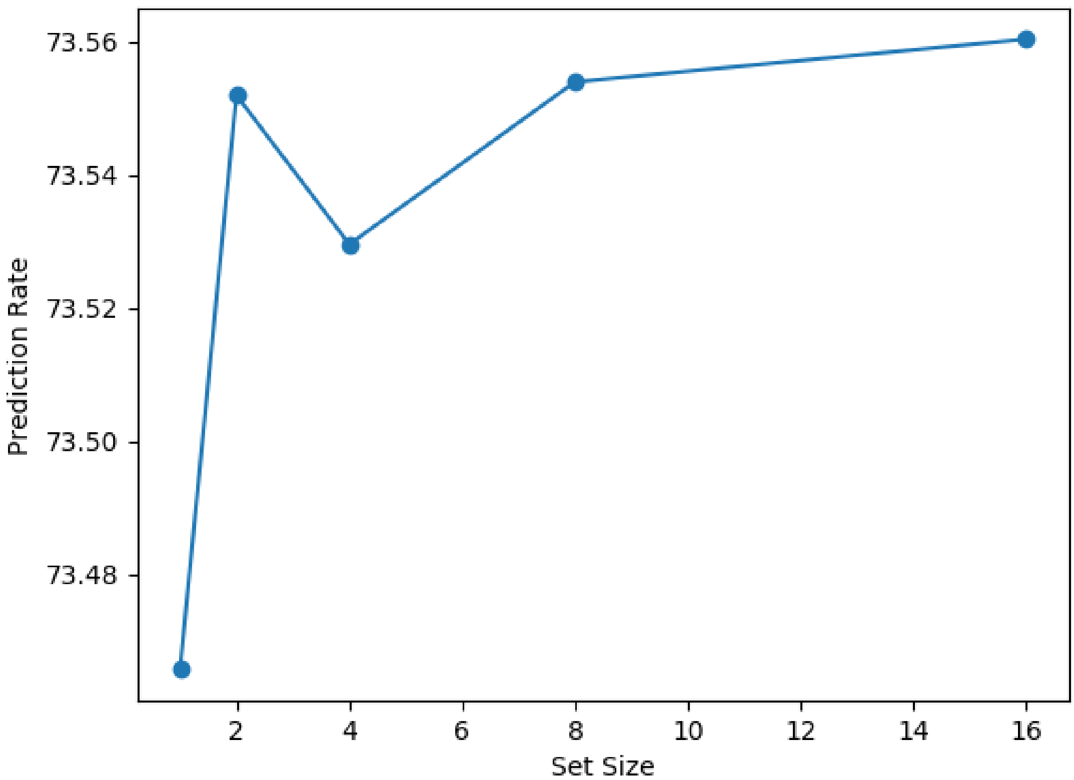
Prediction Rate vs History Size:



Prediction Rate vs Range Size:

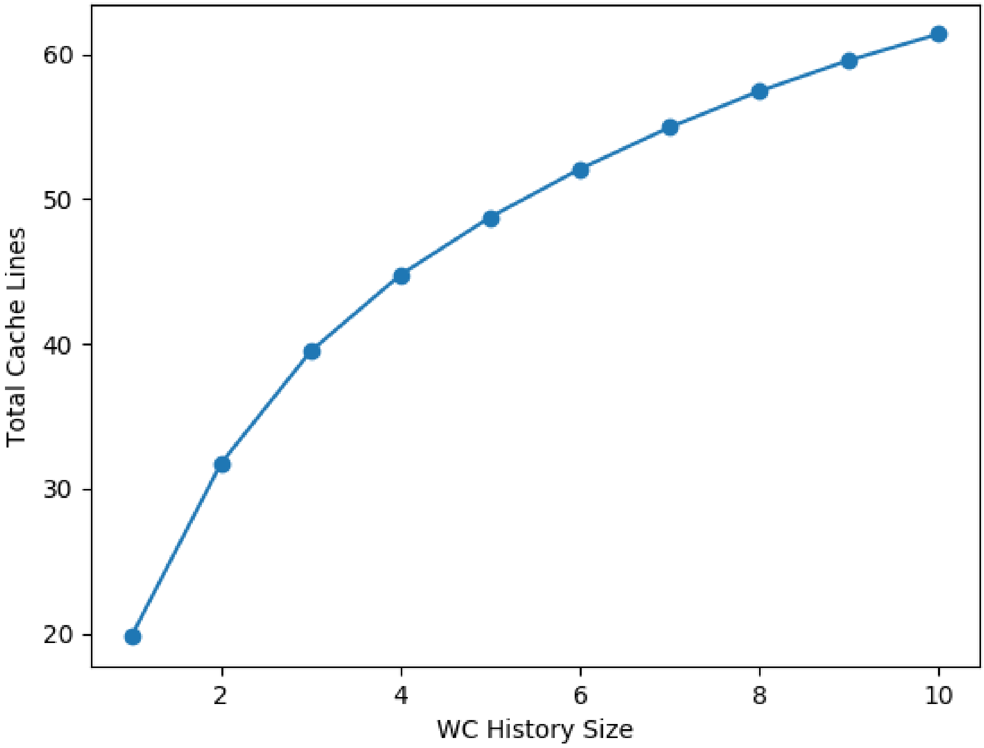


Prediction Rate vs Set Size:

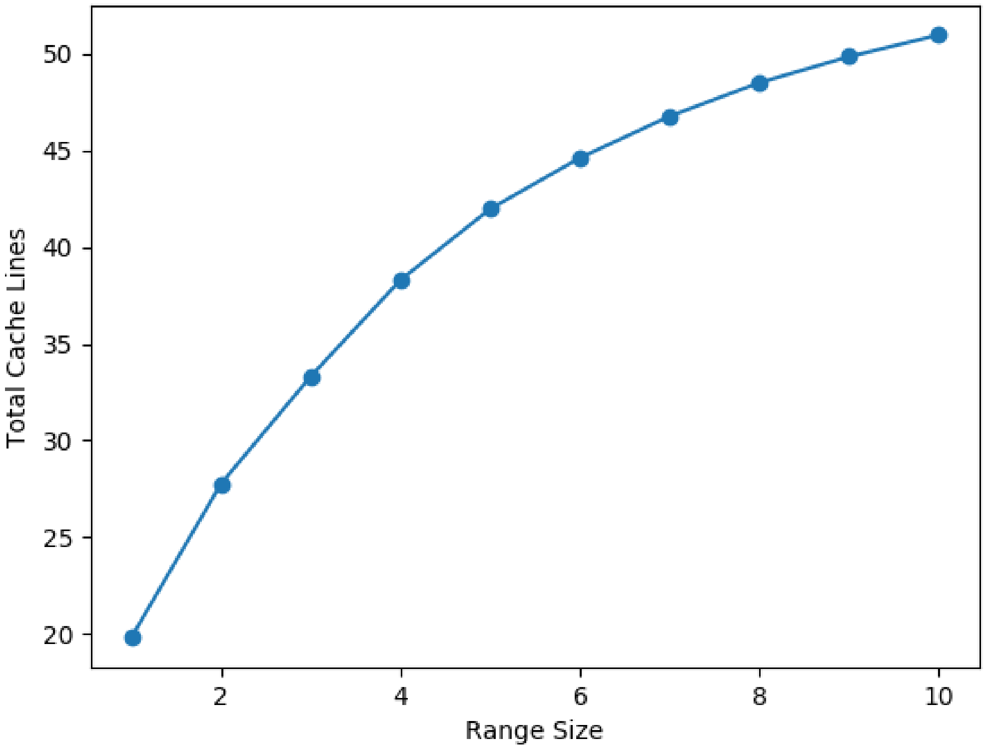


Embarrassingly Parallel

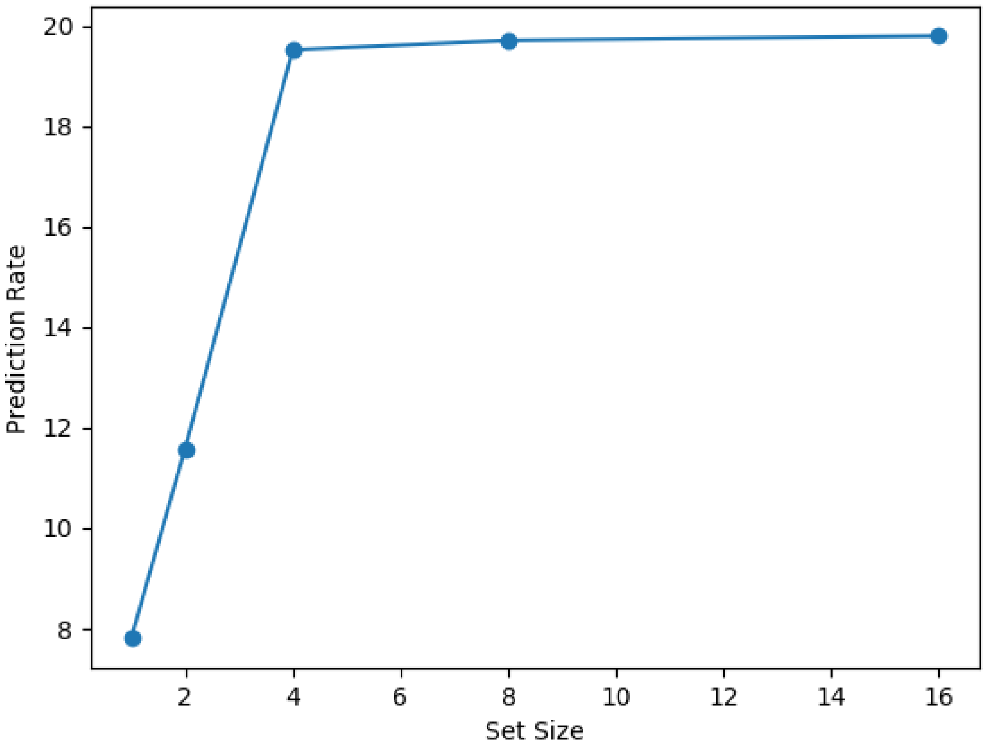
Prediction Rate vs History Size:



Prediction Rate vs Range Size:

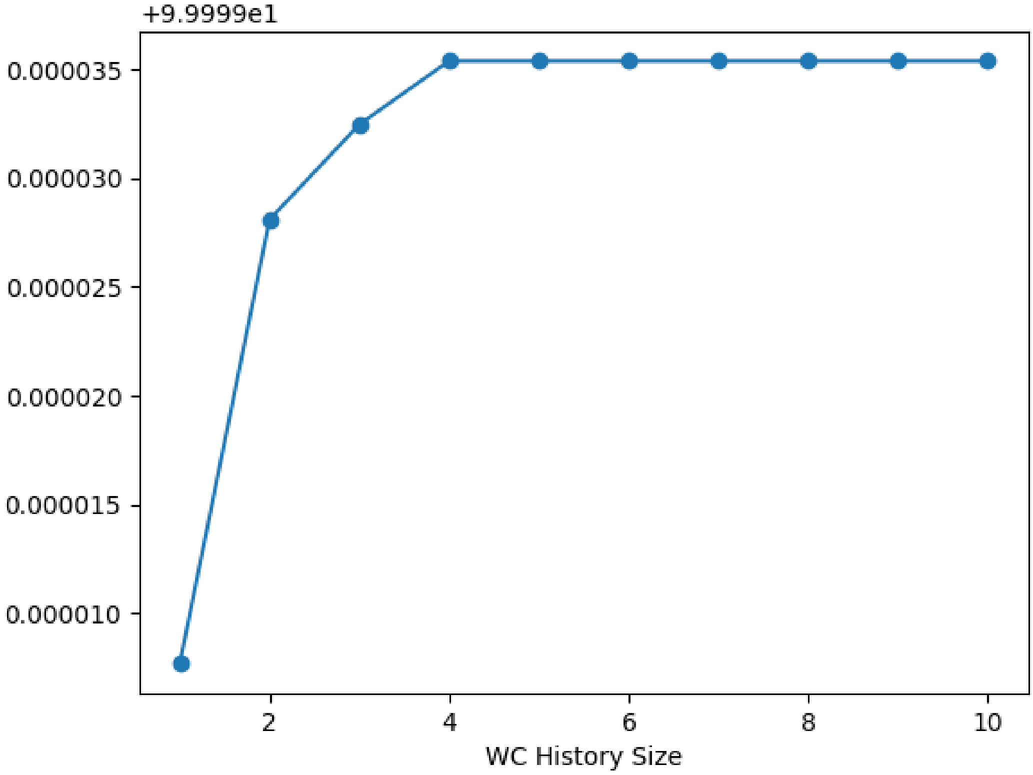


Prediction Rate vs Set Size:

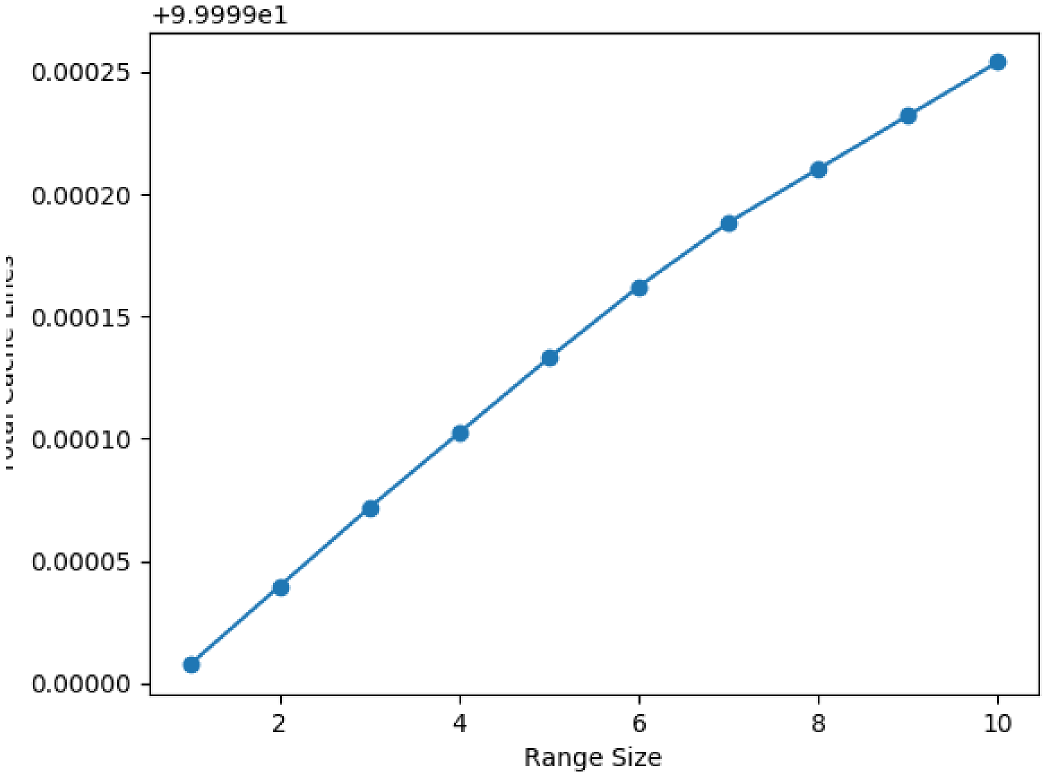


Multi-Grid

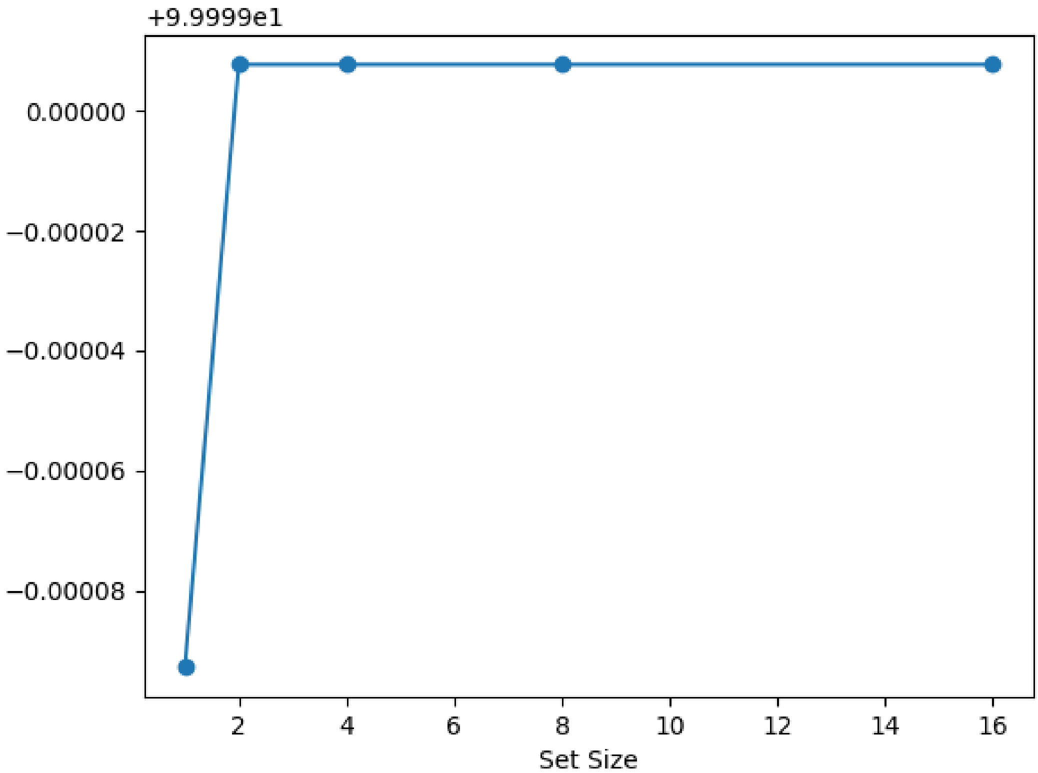
Prediction Rate vs History Size:



Prediction Rate vs Range Size:



Prediction Rate vs Set Size:



Analysis

Conclusion