Cache Modelling

Euan Scott-Watson

November 17, 2021

1 Question 1

I created the simulation in Python. Please see below the time/hit-ratio graph for the FIFO policy cache with m=1024 and n=65536; tested with 3 separate time steps.

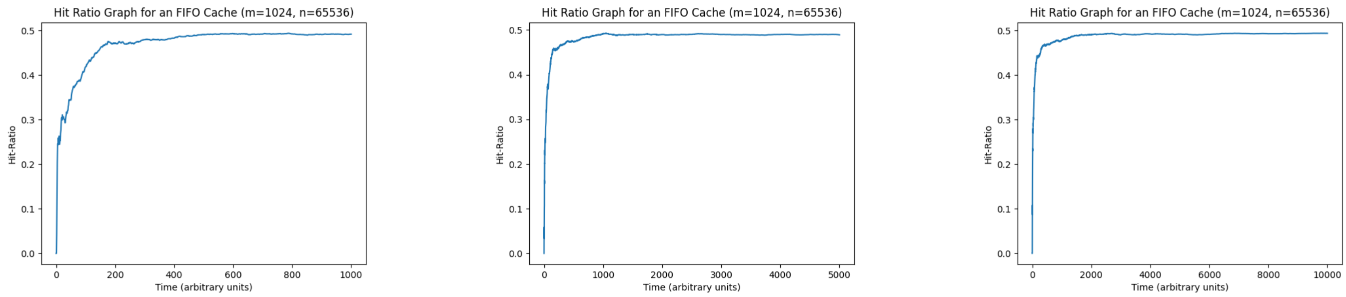


Figure 1: Graphs for times of 1,000, 5,000 and 10,000 arbitrary time units

From my simulations I determined that the hit ratio for a FIFO Cache was ~ 0.49 (and ~ 0.53 for LRU). Therefore, the throughput of my cache to the storage device is `1 - Hit Ratio`, i.e. ~ 0.51 (and ~ 0.47 for LRU)
My estimates are unbiased as I have run multiple tests, all converging to the same values across the simulations. Moreover, the events are all independent of each other causing no bias in picking the timings for each element.
I have modelled the Poisson processes using the Inverse Transform method: $t_k = \frac{-ln(U(0,1))}{\lambda_k}$ where $\lambda_k = \frac{1}{k+1}$.
I filled a diary with timings for each process and iteratively took the one with the smallest time, fetched it from the cache, then re-calculated it's inter-arrival time and put it back into the diary. I continuously did this until the current time, `t`, had reached the time limit, `T`. I took regular readings of the number of hits and misses every 25 arbitrary time units to plot the graphs above with. See the code in **Appendix**.

2 Question 2

$$\text{S} = \{(0,1),(0,2),(1,0),(1,2),(2,0),(2,1)\}$$

$$\text{Q} = \begin{pmatrix} -(\lambda_1+\lambda_2) & 0 & \lambda_1 & 0 & \lambda_2 & 0 \\ 0 & -(\lambda_1+\lambda_2) & \lambda_1 & 0 & \lambda_2 & 0 \\ \lambda_0 & 0 & -(\lambda_0+\lambda_2) & 0 & 0 & \lambda_2 \\ \lambda_0 & 0 & 0 & -(\lambda_0+\lambda_2) & 0 & \lambda_2 \\ 0 & \lambda_0 & 0 & \lambda_1 & -(\lambda_0+\lambda_1) & 0 \\ 0 & \lambda_0 & 0 & \lambda_1 & 0 & -(\lambda_0+\lambda_1) \end{pmatrix}$$

3 Question 3

The total rate out of state s_i is equal to the negative sum of all the rates of reachable states - not including `i`, i.e. $s_i = -\sum_{j\neq i}^{m} \lambda_j \ \forall Q_{i,j} \neq 0$

4 Question 4

$\lambda_0 = 1; \lambda_1 = 1/2; \lambda_2 = 1/3$

$\mathbf{p}Q_c = 0_c$

$$\mathbf{Q} = \begin{pmatrix} -5/6 & 0 & 1/2 & 0 & 1/3 & 0 \\ 0 & -5/6 & 1/2 & 0 & 1/3 & 0 \\ 1 & 0 & -4/3 & 0 & 0 & 1/3 \\ 1 & 0 & 0 & -4/3 & 0 & 1/3 \\ 0 & 1 & 0 & 1/2 & -3/2 & 0 \\ 0 & 1 & 0 & 1/2 & 0 & -3/2) \end{pmatrix}$$

$\mathbf{p} = 0_3 Q_3^{-1}$

$$\mathbf{p} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} -5/6 & 0 & 1 & 0 & 1/3 & 0 \\ 0 & -5/6 & 1 & 0 & 1/3 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1/3 \\ 1 & 0 & 1 & -4/3 & 0 & 1/3 \\ 0 & 1 & 1 & 1/2 & -3/2 & 0 \\ 0 & 1 & 1 & 1/2 & 0 & -3/2) \end{pmatrix}^{-1}$$

$\mathbf{p} = \begin{pmatrix} 18/55 & 12/55 & 0.204 & 0.068 & 4/33 & 2/33 \end{pmatrix}$

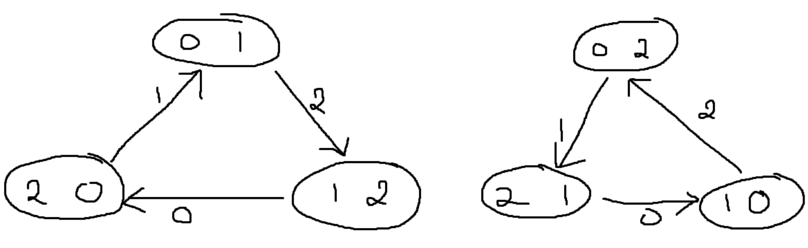
Sum of rates $= \frac{11}{6}$
p(0 or 1) $= \frac{9}{11}$
p(0 or 2) $= \frac{8}{11}$
p(1 or 2) $= \frac{5}{11}$

The hit ratio is calculated by multiplying the probability of a state by the probability of getting one of the elements in that state as the next element, i.e. p((a, b))p(a or b): $(\frac{18}{55} \times \frac{9}{11}) + (\frac{12}{55} \times \frac{8}{11}) + (0.204 \times \frac{9}{11}) + (0.068 \times \frac{5}{11}) + (\frac{4}{33} \times \frac{8}{11}) + (\frac{2}{33} \times \frac{5}{11}) =$ 0.740 which is similar to the value of 0.721 I got from the simulation.

5 Question 5

$$\mathrm{Q} = \begin{pmatrix} -1/3 & 0 & 0 & 1/3 & 0 & 0 \\ 0 & -1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 1/3 & -1/3 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 1/2 & 0 & 0 & 0 & -1/2 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{pmatrix}$$

Only taking into consideration those reachable from the initial configuration of (0, 1):



We can only reach the states: (0, 1), (2, 0), (1, 2), thus:

$$\mathrm{Q} = \begin{pmatrix} -1/3 & 0 & 1/3 \\ 1/2 & -1/2 & 0 \\ 0 & 1 & -1 \end{pmatrix}$$

$\mathbf{p} = 0_3 Q_3^{-1}$

$$\mathbf{p} = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} -1/3 & 0 & 1 \\ 1/2 & -1/2 & 1 \\ 0 & 1 & 1 \end{pmatrix}^{-1}$$

$\mathbf{p} = \begin{pmatrix} 1/2 & 1/3 & 1/6 \end{pmatrix}$

The hit ratio is calculated same as Q4: $(\frac{1}{2} \times \frac{9}{11}) + (\frac{1}{6} \times \frac{5}{11}) + (\frac{1}{3} \times \frac{8}{11}) = \frac{8}{11} = 0.727$, which is similar to the value of 0.743 I got from the simulation.

6 Appendix

6.1 Cache.py Code

```python
class Cache:
    def __init__(self, capacity, starting_values=None):
        self.name = "Cache"
        self.capacity = capacity
        self.hits = 0
        self.misses = 0
        if starting_values:
            self.cache = starting_values + [-1 for _ in range(self.capacity - len(starting_values))]
        else:
            self.cache = [-1 for _ in range(self.capacity)]

    def evict(self, value):
        pass

    def hit(self, value):
        pass

    def get(self, value):
        if value in self.cache:
            self.hits += 1
            self.hit(value)
            return value
        else:
            self.misses += 1
            self.evict(value)
```

6.2 FIFOCache.py Code

```python
from Cache import Cache

class FIFOCache(Cache):
    def __init__(self, capacity, starting_values=None):
        super().__init__(capacity, starting_values)
        self.name = "FIFO"

    def evict(self, value):
        self.cache = self.cache[1:] + [value]
```

6.3 LRUCache.py Code

```python
from Cache import Cache

class LRUCache(Cache):
    def __init__(self, capacity, starting_values=None):
        super().__init__(capacity, starting_values)
        self.name = "LRU"

    def evict(self, value):
        self.cache = [value] + self.cache[:-1]

    def hit(self, value):
        index = self.cache.index(value)
        self.cache = [value] + self.cache[:index] + self.cache[index + 1:]
```

6.4 model.py Code

```python
import sys
import time
from Cache import Cache
from LRUCache import LRUCache
from FIFOCache import FIFOCache
import random, math, bisect

import matplotlib.pyplot as plt


# Functions for sampling
def rate(k):
    return 1 / (k + 1)


def inverse_transform(k):
    l = rate(k)
    U = random.uniform(0, 1)
    return -math.log(U) / l


# Get a start time for all potential ks
def sample_start(n):
    timings = []
    for k in range(n):
        t = inverse_transform(k)
        timings.append((t, k))
    return sorted(timings, key=lambda x: x[0])


def plot_and_save(time_points, hit_ratio):
    # Plots the graph and saves it as a png
    ax = plt.axes()
    ax.plot(time_points, hit_ratio)
    plt.xlabel("Time (arbitrary units)")
    plt.ylabel("Hit-Ratio")
    plt.title("Hit Ratio Graph for an {0} Cache (m={1}, n={2})".format(cache.name, m, n))
    plt.savefig("graphs/{0}_{1}_{2}_{3}.png".format(m, n, T, cache.name))
    plt.show()


def model(n: int, cache: Cache, T: int):
    start = time.time()

    # Arrays for storing data that will be plotted
    hit_ratio = [0]
    time_points = [0]
    # Variable to keep track of current time in simulation
    t = 0

    # Variables for how often to take measurements of HR
    rate_of_data_collection = 1
    next_data_collected = rate_of_data_collection

    diary = sample_start(n)
    while t < T:
        # Get the next entry in the diary
        event = diary.pop(0)
        t = event[0]
        k = event[1]
        cache.get(k)
        tP = t + inverse_transform(k)
        bisect.insort(diary, (tP, k))

        # If enough time has passed, collect more graphing data
        if t // next_data_collected >= 1:
            time_points.append(t)
            hit_ratio.append(cache.hits / (cache.hits + cache.misses))
            next_data_collected += rate_of_data_collection

        # Print the progress, flushing out the previous print
        print("%.2f percent complete - %.2f seconds elapsed\t Current HR: %.2f" % (
            round(t / T * 100, 3),
            round(time.time() - start, 3),
            round(cache.hits / (cache.hits + cache.misses), 3)
        ), end='\r')
        sys.stdout.flush()

```

```python
    print("")
    plot_and_save(time_points, hit_ratio)


if __name__ == "__main__":
    m, n, T, cacheStr = int(sys.argv[1]), int(sys.argv[2]), int(sys.argv[3]), sys.argv[4]
    if cacheStr == "LRU":
        cache = LRUCache(m)
    elif cacheStr == "FIFO":
        cache = FIFOCache(m)
    else:
        Exception()

    model(n, cache, T)
```