Simulation and Modelling
Assessed Coursework
Cache Modelling

Issue date: 20th October
Working: Individual or pairs
Submission deadline: 13th November
Page limit: TWO sheets of A4 plus separate Appendix

1 The Problem

The goal of this coursework is to learn how to model and simulate cache replacement policies commonly used in computer systems. Caches are nowadays ubiquitous, from computer architectures to web proxies, from storage devices to memory caching servers (memcached-based). Their aim is to store popular data items to reduce their access latency. Indeed, when a request finds an item in a cache (*cache hit*) its access latency can be orders of magnitude smaller than in the opposite case of a retrieval from a storage device (*cache miss*).

Caching relies on the assumption that a small fraction of the items accounts for a large fraction of the accesses. Thus the capacity m of a cache is generally much smaller than the total number of items n that can be accessed (including those not in cache). A *cache replacement policy* is a rule that decides which item in the cache should be evicted to allow the cache to store a new item.

The goal of this coursework is to use simulation and continuous-time Markov chains (CTMCs) to compare the performance of two replacement policies: random eviction (RAND) and first-in first-out eviction (FIFO). The reference performance metric will be the *hit ratio*, p_{hit}, which is the proportion of requests that result in a cache hit or, equivalently, the probability that a requested item is in the cache.

2 Reference model

Consider a cache with a fixed capacity of $m \geq 1$ items. Let us assume that the items that the cache can store have all identical sizes and are chosen from a fixed population of $n > m$ items labelled by the integers $1, \ldots, n$. An item can either be outside the cache or occupy a single position in the cache.

At time $t = 0$ assume that the cache is pre-loaded with the items labelled $i = 1, \ldots, m$ respectively in positions $1, \ldots, m$ in the cache. Subsequently, users' requests for item k arrive according to a Poisson process with some specified rate $\lambda_k, 1 \leq k \leq n$.

When a request arrives, if the requested item is in the cache we have a *cache hit*, otherwise it is a *cache miss*. The cache replacement policy takes care of two duties:

- If a request for item k yields a cache miss, the policy selects a *victim item* among those present in the cache, and evicts it to free space for item k. During eviction, the policy may rearrange some or all of the items already in the cache.
- Upon a cache hit, the policy does not evict items.

The replacement policies you need to consider work as follows:

FIFO policy specification:

Cache miss: The item in position 1 is evicted. Items in position $2, \ldots, m$ are shifted into positions $1, \ldots, m-1$ respectively. The requested item k is then placed in position m.

Cache hit: no changes are made to the cache.

RAND policy specification:

Cache miss: the victim item is chosen randomly from the positions $1, \ldots, m$. Item k is added at the chosen location, evicting the item that was cached there.

Cache hit: no changes are made to the cache.

3 What to do

Assume that item k is requested at Poisson rate λ_k.

Q1. Using a language of your choice, implement a simulator for the RAND and FIFO policies for given values of m and n and with $\lambda_k = 1/k, 1 \leq k \leq n$ (item 1 is the most popular and item n the least). Initially, pre-load the cache with items $1, \ldots, m$ as described above.

Compute point and interval estimates for the hit ratio and the cache *miss rate*, i.e. the throughput of request traffic from the cache to the storage device (see below), for $n = 1000$ and $m = 10, 50$ and 100 for both FIFO and RAND. Do you notice anything? (You can cheat by reading on!)

Note that the cache miss rate can be computed from $(1 - p_{hit}) \sum_k \lambda_k$ or from M/T where M is the absolute number of misses and T the total simulation time. You should report the latter.

In addition to these estimates, explain *briefly* how you have chosen to model the n Poisson processes, including an explanation of how you advance time and how you choose the next item to be requested. Please attach your code as an Appendix.

Q2. You are now going to model the cache as a CTMC. The states of the CTMC should be a vector of m integers, $(s_1, \ldots, s_m)$ where s_i is the label of the item in position i of the cache. For example, for $m = 2$ and $n = 3$ the state $(3, 2)$ represents the situation where item 3 occupies position 1 and item 2 occupies position 2; item 1 is therefore uncached.

Assume that item k is requested at an arbitrary given Poisson rate $\lambda_k, 1 \leq k \leq n$. For $m = 2$ and $n = 3$, write down the set of possible states, S say, and define the generator matrix of the CTMC in terms of the λ_k, assuming the RAND replacement policy.

Q3. For arbitrary m and $n \geq m$, what will be the total rate out of state $(s_1, \ldots, s_m)$? Explain briefly your answer.

Q4. Now set $\lambda_k = 1/k$ (as in the simulator) and $m = 2, n = 3$. By solving the global balance equations, compute the probability of each state of the cache under RAND and use the result to calculate the hit ratio. Does this tally with your simulation when you set $m = 2, n = 3$? (It should!)

Q5. For $m = 2, n = 3$ construct the generator matrix for FIFO replacement and show that FIFO has the same hit ratio as RAND. You should find that your CTMC is *not* irreducible so you should consider only states that are reachable from the initial configuration at time $t = 0$. As before, assume $\lambda_k = 1/k$.

We anticipate a relative weighting of 40%, 20%, 10%, 20% and 10% for the five questions.