EC504 Advanced Data Structure HW0

Simple Caching Report

Student: Jiaxing TIAN (98336741)

3.1 Prediction Schemes

Scheme 1

1. In the uniformly randomly interleaved stream, if data block x_i is requested by the user, then fetch and store the data block $x_i + 1$ in the cache. After the block is delivered from cache, the data block is evicted.

Examples:

(1) For one data stream, 1, 2, 3, 4… when, block 1 is requested by the user, block 2 is cached, when block 2 is requested and delivered from cache, block 3 is cached; when block 3 is requested, and delivered from the cache, block 4 is cached; when block 4 is requested, block 5 should be cached.

(2) For two stream 1, 2, 3, 4… and 10, 11, 12... In the stream 1, 2, 10, 3, 11, 12, 4… which is the combination of two streams. When 1 is requested by the user, block 2 is cached, when block 2 is requested and served, block 3 is cached, when block 10 is requested, block 11 is cached, when 3 is requested and served, block 4 is cached, when 11 is served, 12 is cached, when 12 is requested and served, 13 is cached, when block 4 is requested, block 5 is cached.

(3) For data stream 1, 2, 3; 7, 8, 9 and 20, 21, 22, the combination can be any form, such as 1, 20, 2, 7, 8, 21, 22, 3, 9. When 1 is requested by the user, cache block 2, when 20 is requested, cache 21; when 2 is requested and delivered from cache, cache block 3; when 7 is requested, cache 8; when 8 is delivered cache 9; when 21 is requested, cache 22; when 22 is delivered cache 23; when 3 is requested, cache block 4; when 9 is requested cache 10.

Scheme 2

2. In the uniformly randomly interleaved stream, if data block x_i and block $x_i + 1$ is requested by the user, $x_i + 2$ is cached and then start the cache schemes. When the cache scheme is started, if x_k is cached previously and requested by the user, fetch and store

the data block $x_k + 1$ in the cache. After the block is delivered from cache, the data block is evicted.

Examples:

(1) For single data stream, 1, 2, 3, 4, 5; when block 1 and block 2 is requested by the user, the cache scheme is started, and block 3 is cached; when block 3 is requested and delivered from the cache, block 4 is cached; when block 4 is requested and served from the cache, cache block 5; when block 5 is requested, cache 6.
(2) As for data stream 1, 2, 3, 4, and 10, 11, 12, the combination is 1, 2, 10, 3, 11, 12, 4… in this case, after 1, 2 is requested by the user, the cache start to fetch blocks. Block 3 is cached; after block 10 is requested, block 3 is requested and delivered from cache, then cache 4. After 11 is requested, because 10 is requested before, block 12 will be cached; after 12 is requested, block 13 is cached. When 4 is served, block 5 will be cached.
(3) For data stream 1, 2, 3; 7, 8, 9 and 20, 21, 22, the combination can be any form, such as 1, 20, 2, 7, 8, 21, 22, 3, 9. 1 is requested, 20 is requested, 2 is requested, cache 3, 7 is requested, 8 is requested, cache 9, 21 is requested, cache 22; 22 is requested, cache 23; 3 is requested from cache, cache 4; 9 is requested and delivered from cache, cache 10;

Scheme 3

3. In the uniformly randomly interleaved stream, if data block x_i, x_i+1, and x_i +2 are requested, then x_i +3 is cached and the cache scheme is started. After the cache scheme is started, if x_k is cached previously and requested, then x_k+1 is cached. After the block is delivered from cache, the data block is evicted.

Examples:

(1) For single data stream, 1, 2, 3, 4, 5. The cache scheme start when 1, 2, 3 are all requested from the user, and block 4 is cached; when block is requested and delivered from the cache, block 5 is cached. When block 5 is requested, cache block 6.
(2) As for data stream 1, 2, 3, 4, and 10, 11, 12, 13 the combination is 1, 2, 10, 3, 11, 12, 4, 13. 1 is requested, 2 is requested, 10 is requested, 3 is requested, cache 4; 11 is requested, 12 is requested, cache 13; 4 is requested and delivered by cache, cache 5; 13 is requested, cache 14.
(3) For data stream 1, 2, 3; 7, 8, 9 and 20, 21, 22, the combination can be any form, such as 1, 20, 2, 7, 8, 21, 22, 3, 9. 1 is requested, 20 is requested, 2 is requested, 7 is requested, 8 is requested, 21 is requested, 22 is requested, cache 23; 3 is requested, cache 4; 9 is requested, cache 10;

Scheme 4

4. In the uniformly randomly interleaved stream, the data block x_i is cached, only if data block x_i -1 is requested by the user and not cached previous in the cache. After the block is delivered from cache, the data block is evicted.

Examples:

(1) For single stream 1, 2, 3, 4, 5. 1 is requested and not cached before, cache 2. 2 is requested; 3 is requested but not cached, cache 4; 4 is requested; 5 is requested and not cached, cache 6.
(2) As for data stream 1, 2, 3, 4, and 10, 11, 12, 13 the combination is 1, 2, 10, 3, 11, 12, 4, 13. 1 is requested but not cached, cache 2; 2 is requested, 10 is requested but not cached, cache 11, 3 is requested and not cached, cache 4; 11 is requested and delivered from cache, 12 is requested and not cached, cache 13; 4 is requested and served from cache; 13 is requested and served from the cache.
(3) For a single stream, if the user want to go back during play the video, the block sequence may look like 20, 21, 22, 23, 7, 8, 9; in this case, 20 is requested and not cached, cache 21; 21 is requested; 22 is requested and not cached, cache 23; 23 is requested; 7 is requested and not cached, cache 8; 8 is requested and delivered from cache; 9 is requested and not cached, cache 10.

Scheme 5

5. In the uniformly randomly interleaved stream, cache a random data block when block x_i is requested from the user. After the block is delivered from cache, the data block is evicted.

Examples:

(1) For single stream 1, 2, 3, 4. 1 is requested, cache a random block 7, 2 is requested cache random block 10, 3 is requested, cache random block 21; 4 is requested, cache random block 6.
(2) As for data stream 1, 2, 3, and 10, 11, 12 the combination is 1, 2, 10, 3, 11, 12; 1 is requested, cache random block 5; 2 requested, cache 9; 10 requested, cache 14; 3 requested, cache 25; 11 requested, cache 19; 12 requested, cache 41;
(3) As for play back stream, 20, 21, 22, 7, 8, 9; 20 requested, cache 2; 21 requested cache 31; 22 requested, cache 10; 7 requested cache 25; 8 requested cache 45; 9 requested, cache 11.

3.2 Prediction performance

Cache-frac is the ration of data block delivered from cache to the total requested data block.

Cache-frac = Block from cache/total data block

3.3 Analyze prediction schemes

For one stream in the uniformly randomly interleaved stream x_0, x_1, x_3…, x_n, X= {x_i:$xi \epsilon A$}. Consider x_i as a random variable, then sample space is A. Consider data block delivered from cache as a success event.

(1) **Scheme1:** For one stream in the uniformly randomly interleaved stream x_0, x_1, x_3…, x_n, X= {x_i:$xi \epsilon A$}. Consider x_i as a random variable, then sample space is A. For data block x_i, if x_i is requested, then x_i + 1 is cached. Consider data block requested and delivered from cache as a success event, then after random block $x_{(i-1)}$ (i>=1) is requested, if the next requested block from this stream is $x_i = x_{(i-1)}$ +1, then the event success; otherwise the event fail. After the first request is made, the number of success event construct a Binomial Distribution. The probability for the event to be success is p; and the probability for the event to be failed is 1-p. Then probability for k success event is shown as:

$$p(k) = \binom{k}{n} p^k (1-p)^{n-k}$$

The expected success event is shown as E(x)=np; the Cache-frac = np/(n+1); when n is relatively large, it becomes <u>Cache-frac = p</u>; the case is the same for m stream in the uniformly randomly interleaved stream.

(2) **Scheme 2:** For one stream in the uniformly randomly interleaved stream x_0, x_1, x_3…, x_n, X= {x_i:$xi \epsilon A$}. Consider x_i as a random variable, then sample space is A. Consider data block delivered from cache as a success event. For data block x_{i-2} (i>=2) that is not cached before, if $x_{i-1} = x_{i-2}$ +1 and $x_i= x_{i-1}$ +1 then x_i is cached and delivered, which means event success for random variable x_i. Otherwise the event fail. After the first 2 request is made, the number of success event construct a Binomial Distribution. For data block x_{i-2} that is cached before, if $x_{i-1} = x_{i-2}$ +1 and $x_i= x_{i-1}$ +1 then x_i is cached and delivered, which also means event success for random variable x_i. Otherwise the event fail. For The probability for the event x_i to be success is p^2. The probability for the event to be fail is $1-p^2$; Then probability for k success event is shown as:

$$p(k) = \binom{k}{n-1} p^{2k}(1-p^2)^{n-1-k}$$

The expected success event is shown as E(x)=(n-1)p^2; the Cache-frac = (n-1)p^2/(n+1); when n is relatively large, it becomes <u>Cache-frac = p^2</u>; the case is the same for m stream in the uniformly randomly interleaved stream.

(3) **Scheme 3:** For one stream in the uniformly randomly interleaved stream x_0, x_1, x_3…, x_n, X= {x_i:$xi \in A$}. Consider x_i as a random variable, then sample space is A. Consider data block delivered from cache as a success event. For data block x_{i-3} (i>=3) that is not cached before, if x_{i-2} = x_{i-1} +1; x_{i-1} = x_{i-2} +1 and x_i= x_{i-1} +1 then x_i is cached and delivered, which means event success for random variable x_i. Otherwise the event fail. After the first 3 request is made, the number of success event construct a Binomial Distribution. For data block x_{i-3} that is cached before, if x_{i-2} = x_{i-1} +1; x_{i-1} = x_{i-2} +1 and x_i= x_{i-1} +1 then x_i is cached and delivered, which also means event success for random variable x_i. Otherwise the event fail. For The probability for the event x_i to be success is p^3. The probability for the event to be fail is 1-p^3; Then probability for k success event is shown as:

$$p(k) = \binom{k}{n-2} p^{3k}(1-p^3)^{n-2-k}$$

The expected success event is shown as E(x)=(n-2)p^3; the Cache-frac = (n-2)p^3/(n+1); when n is relatively large, it becomes <u>Cache-frac = p^3</u>; the case is the same for m stream in the uniformly randomly interleaved stream.

(4) **Scheme 4:** For one stream in the uniformly randomly interleaved stream x_0, x_1, x_3…, x_n, X= {x_i:$xi \in A$}. Consider x_i as a random variable, then sample space is A. Consider data block delivered from cache as a success event. If data block $x_{i-1} \neq x_{i-2}$ +1 and x_i =x_{i-1}+1, (i>=2) then x_i is cached and delivered, which means event success for random variable x_i. Otherwise the event fail. The probability for the event x_i to be success is (1-p)*p. The probability for the event to be fail is 1-(p-p^2); Then probability for k success event is shown as:

$$p(k) = \binom{k}{n-1} (p-p^2)^k(1-p+p^2)^{n-1-k}$$

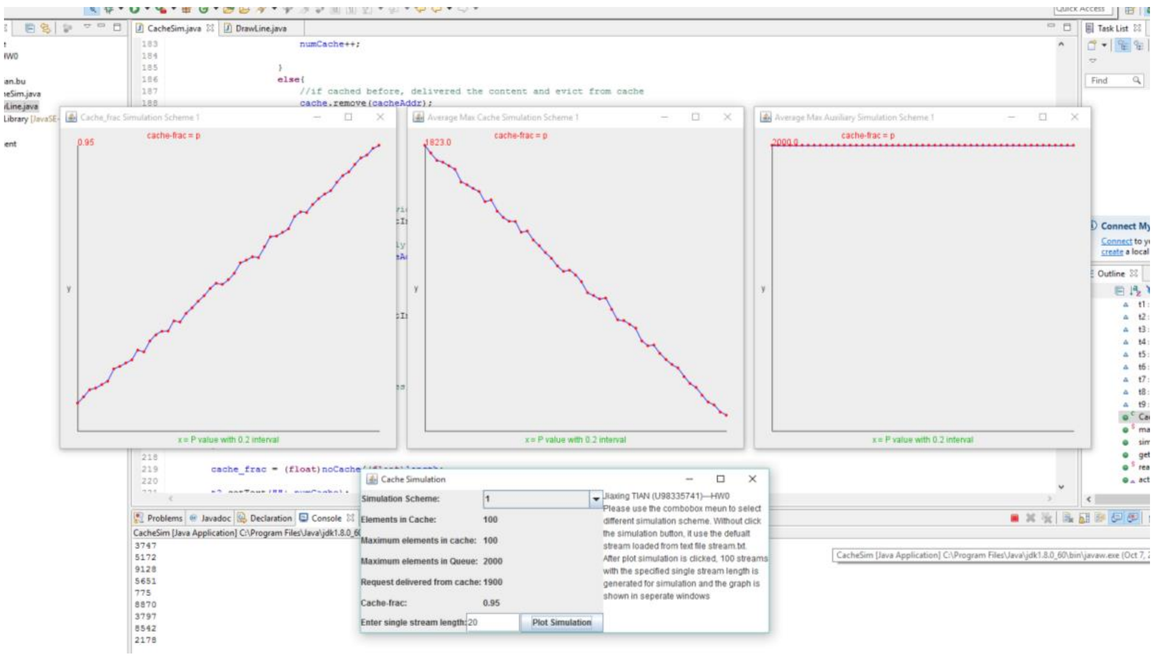
The expected success event is shown as E(x)=(n-1)(p-p^2); the Cache-frac = (n-1)(p-p^2)/(n+1); when n is relatively large, it becomes <u>Cache-frac = p-p^2</u>; When i=1, x_1 can be cached, if x_1=x_0+1. However, when computing the average number and n is large, the effect of x_1 can be ignored. The case is the same for m stream in the uniformly randomly interleaved stream.

(5) **Scheme 5:** For one stream in the uniformly randomly interleaved stream x_0, x_1, x_3…, x_n, X= {x_i:$xi \in A$}. Consider x_i as a random variable, then sample space is A. Consider data block delivered from cache as a success event. After x_i is requested,

cache a random block. The probability of request a cache block for x_i is i/N. For the best case, where all n block is requested and delivered from cache, cache-frac = 1; and the worst case is none of the request is cached, where cache-frac =0. Set k to be the number of block delivered from cache. In average, $E(k) = \frac{n}{N} * n$; Since N is much larger than n, the probability of request the cached block is very little approximately zero. Hence, cache-frac = 0;

3.4 Implement a simulator

The simulator is implemented in Java with eclipse under the project folder CacheStream_HW0. Build the project and run it, the default stream input is a text file under the same folder called "stream.txt". Change the content you want to test the simulation result. The scheme is selected from the combo-box menu. Click plot simulation button, it will generate 100 stream with specified single stream length (by default 20), and return an interleaving stream for simulation and plot graph. The graph is shown in 3 separate windows.



Cache_frac Simulation Scheme 1
cache-frac = p
0.95
y
x = P value with 0.2 interval
Average Max Cache Simulation Scheme 1
cache-frac = p
1823.0
y
x = P value with 0.2 interval
Average Max Auxiliary Simulation Scheme 1
cache-frac = p
2000.0
y
x = P value with 0.2 interval
Cache Simulation
Plot Simulation

```
                numCache++;
            }
            else{
                //if cached before, delivered the content and evict from cache
                cache.remove(cacheAddr);
```

```
        cache_frac = (float)noCache
```

```
CacheSim [Java Application] C:\Program Files\Java\jdk1.8.0_6
3747
5172
9128
5651
775
8870
3797
8542
2178
```

Simulation Scheme: 1
Elements in Cache: 100
Maximum elements in cache: 100
Maximum elements in Queue: 2000
Request delivered from cache: 1900
Cache-frac: 0.95
Enter single stream length: 20

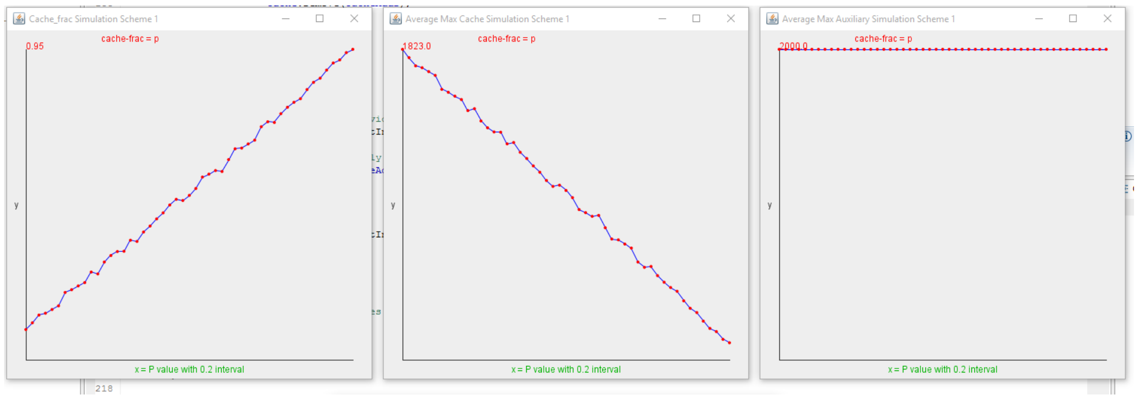
Jiaxing TIAN (U98335741)—HW0
Please use the combobox meun to select different simulation scheme. Without click the simulation button, it use the defualt stream loaded from text file stream.txt.
After plot simulation is clicked, 100 streams with the specified single stream length is generated for simulation and the graph is shown in seperate windows

The x axis is predictability p [0,1] with 0.02 interval. Hence, a total of 51 points are drawn in the pane.

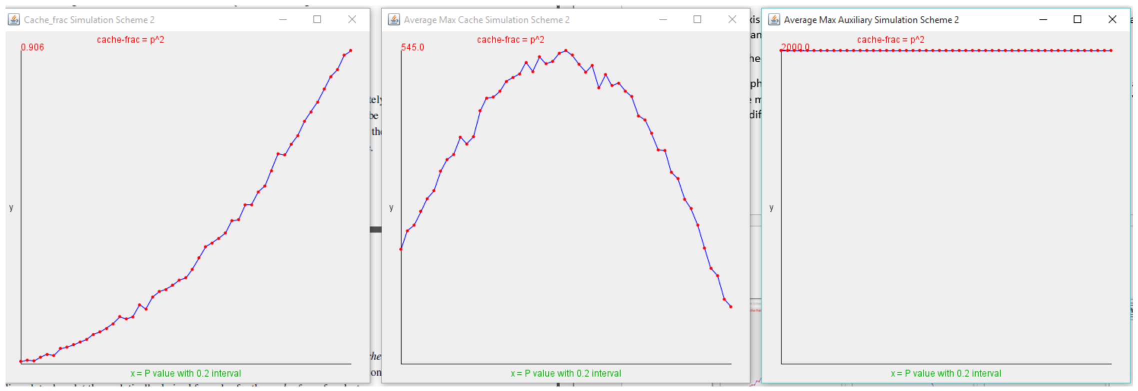
3.5 Cache-frac and size

The graph is plotted in the simulator. Average Cache-frac, average maximum cache and average maximum auxiliary data structure size is plotted in three separate windows. You can modify the length of each single stream to get a more precise graph.

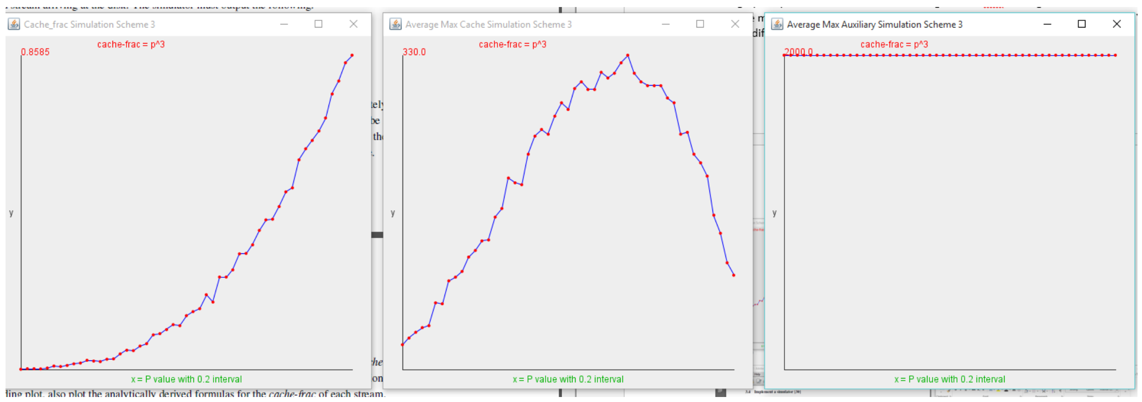
Scheme 1



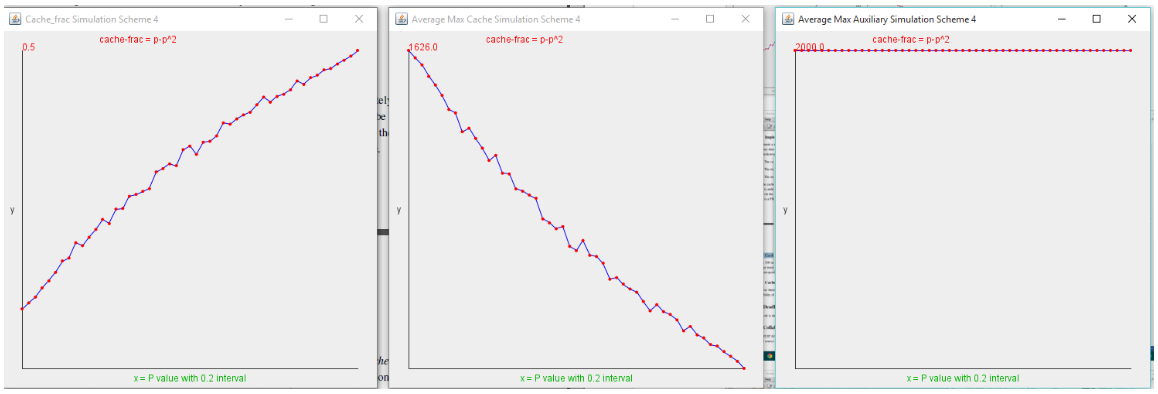
Scheme 2



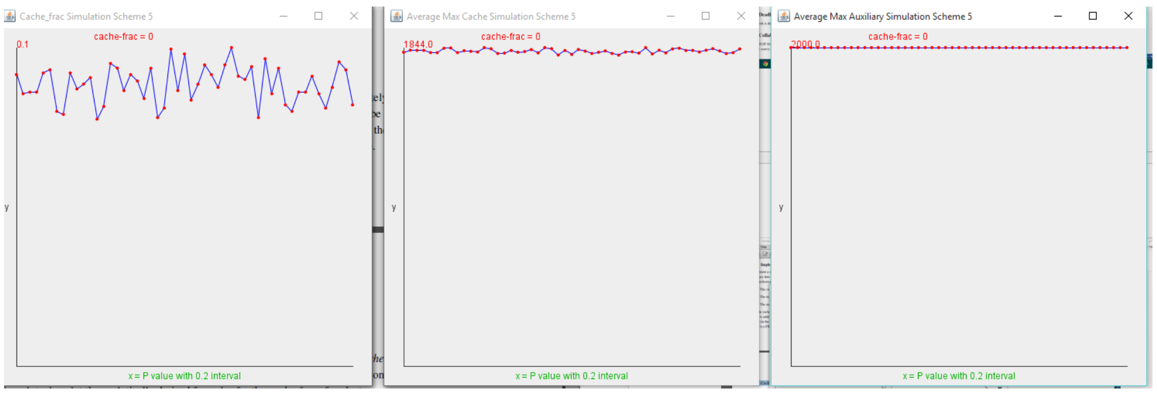
Scheme 3



Scheme 4



Scheme 5



3.6 Cache size

Since the cache size is restricted, set the size of the cache to be S. In that case, if the two consecutive request that comes from a single stream is separated longer than S, the new request cannot be recognized as the consecutive of previous request, since the former request is remove due to size restriction. For m streams, the probability for two request from the same stream arrive k request apart is

$$p(k) = \frac{(m-1)^k}{m^k}\frac{1}{m}$$

In order for the two consecutive request to be recognized, k should be smaller than S. The probability is $P = \sum_{k=0}^{S-1} p(k)$, which is

$$p = 1 - \frac{(m-1)^s}{m^s}$$

Then, 1-p tells the probability a useful data is evicted from cache. 1-p<=e

Hence, we obtain

$$S \geq \frac{lne}{\ln({}^{m-1}/_{m})}$$