Hash Code

Streaming videos

Problem statement for Online Qualification Round, Hash Code 2017

Introduction

Have you ever wondered what happens behind the scenes when you watch a YouTube video? As more and more people watch online videos (and as the size of these videos increases), it is critical that video-serving infrastructure is optimized to handle requests reliably and quickly.

This typically involves putting in place cache servers, which store copies of popular videos. When a user request for a particular video arrives, it can be handled by a cache server close to the user, rather than by a remote data center thousands of kilometers away.

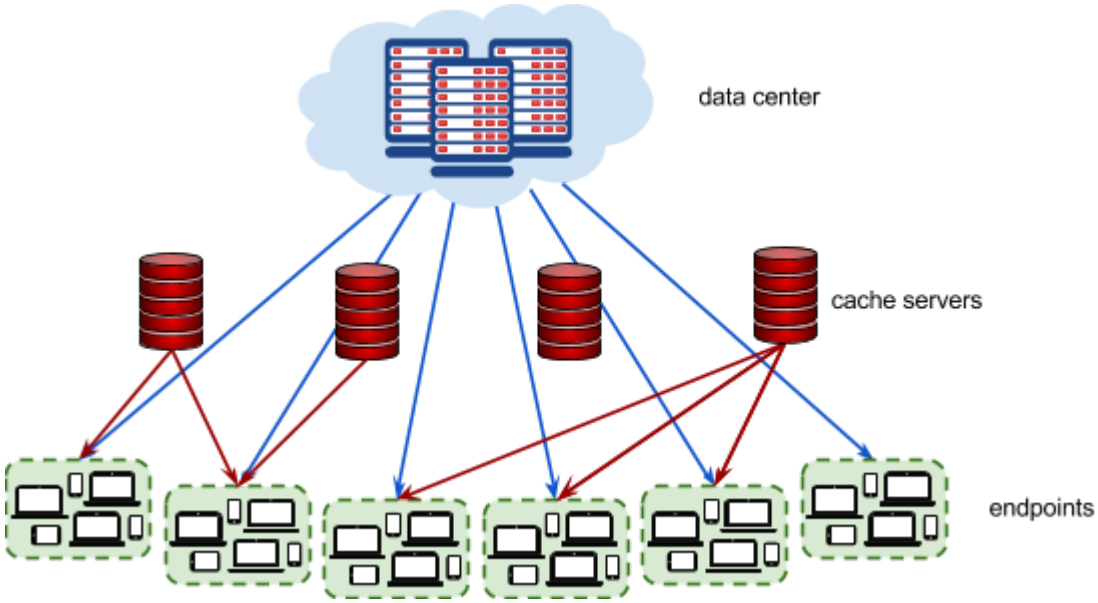
But how should you decide which videos to put in which cache servers?

Task

Given a description of cache servers, network endpoints and videos, along with predicted requests for individual videos, **decide which videos to put in which cache server** in order to minimize the average waiting time for all requests.

Problem description

The picture below represents the video serving network.



Videos

Each video has a size given in megabytes (MB). The data center stores **all videos**. Additionally, each video can be put in 0, 1, or more **cache servers**. Each cache server has a maximum capacity given in megabytes.

Endpoints

Each **endpoint** represents a group of users connecting to the Internet in the same geographical area (for example, a neighborhood in a city). Every endpoint is connected to the data center. Additionally, each endpoint may (but doesn't have to) be connected to 1 or more cache servers**.**

Each endpoint is characterized by the latency of its connection to the data center (how long it takes to serve a video from the data center to a user in this endpoint), and by the latencies to each cache server that the

endpoint is connected to (how long it takes to serve a video stored in the given cache server to a user in this endpoint).

Requests

The predicted **requests** provide data on how many times a particular video is requested from a particular endpoint.

Input data set

The input data is provided as a data set file - a plain text file containing exclusively ASCII characters with a single '\n' character at the end of each line (UNIX-style line endings).

Videos, endpoints and cache servers are referenced by integer IDs. There are ***V*** videos numbered from 0 to $V-1$, E endpoints numbered from 0 to $E-1$ and C cache servers numbered from 0 to $C-1$.

File format

All numbers mentioned in the specification are natural numbers that fit within the indicated ranges. When multiple numbers appear in a single line, they are separated by a single space.

The first line of the input contains the following numbers:

- ***V*** $(1 \leq V \leq 10000)$ - the number of videos
- ***E*** $(1 \leq E \leq 1000)$ - the number of endpoints
- ***R*** $(1 \leq R \leq 1000000)$ - the number of request descriptions
- ***C*** $(1 \leq C \leq 1000)$ - the number of cache servers
- ***X*** $(1 \leq X \leq 500000)$ - the capacity of each cache server in megabytes

The next line contains ***V*** numbers describing the sizes of individual videos in megabytes: $S_0, S_1, \ldots S_{V-1}$.

S_i is the size of video i in megabytes $(1 \leq S_i \leq 1000)$.

The next section describes each of the endpoints one after another, from endpoint 0 to endpoint $E-1$. The description of each endpoint consists of the following lines:

- a line containing two numbers:
  - L_D $(2 \leq L_D \leq 4000)$ - the latency of serving a video request from the data center to this endpoint, in milliseconds
  - ***K*** $(0 \leq K \leq C)$ - the number of cache servers that this endpoint is connected to
- ***K*** lines describing the connections from the endpoint to each of the *K* connected cache servers. Each line contains the following numbers:
  - ***c*** $(0 \leq c < C)$ - the ID of the cache server

- ***L_c*** $(1 \leq L_c \leq 500)$ - the latency of serving a video request from this cache server to this endpoint, in milliseconds. You can assume that latency from the cache is strictly lower than latency from the data center $(1 \leq L_c < L_D)$.

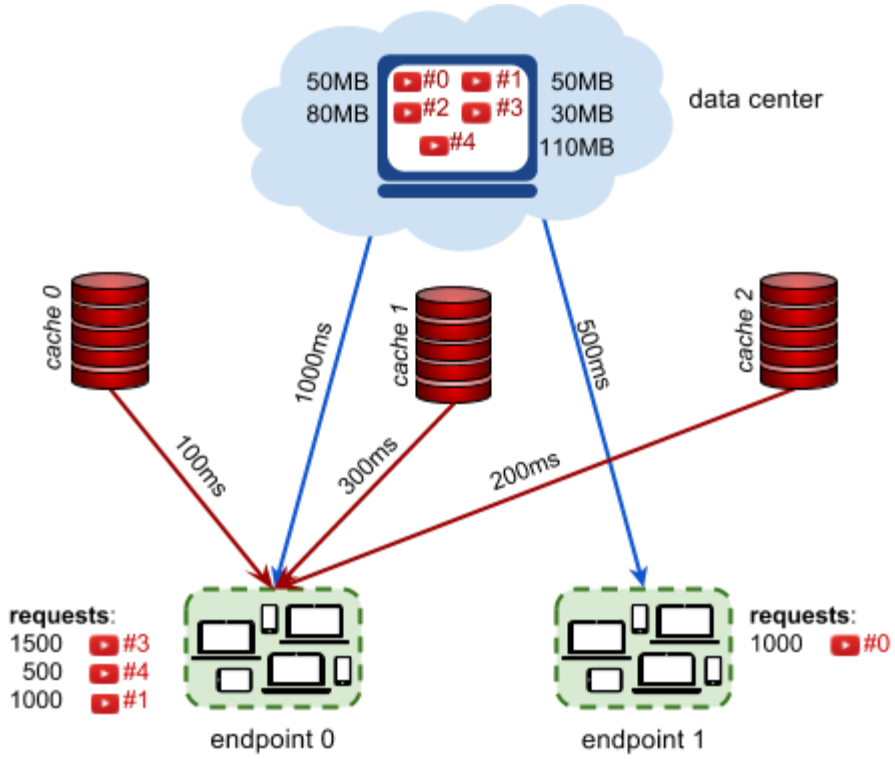
Finally, the last section contains ***R*** request descriptions in separate lines. Each line contains the following numbers:

- ***R_v*** $(0 \leq R_v < V)$ - the ID of the requested video
- ***R_e*** $(0 \leq R_e < E)$ - the ID of the endpoint from which the requests are coming from
- ***R_n*** $(0 < R_n \leq 10000)$ - the number of requests

Example

5 2 4 3 100 50 50 80 30 110 1000 3 0 100 2 200 1 300 500 0 3 0 1500 0 1 1000 4 0 500 1 0 1000	5 videos, 2 endpoints, 4 request descriptions, 3 caches 100MB each. Videos 0, 1, 2, 3, 4 have sizes 50MB, 50MB, 80MB, 30MB, 110MB. Endpoint 0 has 1000ms datacenter latency and is connected to 3 caches: The latency (of endpoint 0) to cache 0 is 100ms. The latency (of endpoint 0) to cache 2 is 200ms. The latency (of endpoint 0) to cache 1 is 300ms. Endpoint 1 has 500ms datacenter latency and is not connected to a cache. 1500 requests for video 3 coming from endpoint 0. 1000 requests for video 0 coming from endpoint 1. 500 requests for video 4 coming from endpoint 0. 1000 requests for video 1 coming from endpoint 0.

Example input file.



Connections and latencies between the endpoints and caches of example input.

Submissions

File format

Your submission should start with a line containing a single number ***N*** ($0 \leq N \leq C$) - the number of cache server descriptions to follow.

Each of the subsequent ***N*** lines should describe the videos cached in a single cache server. It should contain the following numbers:

- ***c*** $(0 \leq c < C)$ - the ID of the cache server being described,
- the IDs of the videos stored in this cache server: $v_0, \ldots, v_n$ $(0 \leq v_i < V)$ (at least 0 and at most V numbers), given in any order without repetitions

Each cache server should be described in at most one line. It is not necessary to describe all cache servers: if a cache does not occur in the submission, this cache server will be considered as empty. Cache servers can be described in any order.

Example

`3`
`0 2`
`1 3 1`
`2 0 1`
	`We are using all 3 cache servers.`
	`Cache server 0 contains only video 2.`
	`Cache server 1 contains videos 3 and 1.`
	`Cache server 2 contains videos 0 and 1.`

Example submission file.

Validation

The output file is valid if it meets the following criteria:

- the format matches the description above
- the total size of videos stored in each cache server does not exceed the maximum cache server capacity

Scoring

The score is the average time saved per request, in microseconds. (Note that the latencies in the input file are given in milliseconds. The score is given in microseconds to provide a better resolution of results.)

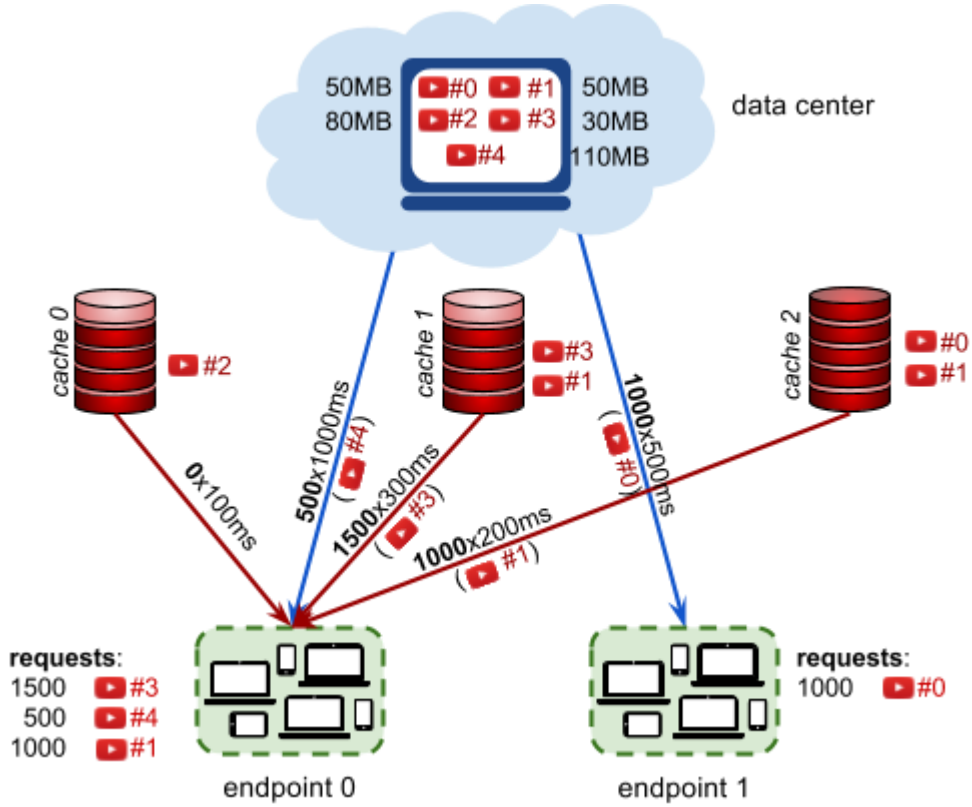
For each request description (R_v, R_e, R_n) in the input file, we choose the best way to stream the video R_v to the endpoint R_e. We pick the lowest possible latency $L = min(L_D, L_0, \ldots, L_{k-1})$, where L_D is the latency of serving a video to the endpoint R_e from the data center, and $L_0, \ldots, L_{k-1}$ are latencies of serving a video to the endpoint R_e from each cache server that:

- is connected to the endpoint R_e, and
- contains the video R_v

The time that was saved for each request is $L_D - L$ (This is the time it would take to stream the video from the data center, minus the time it actually took. If the video is in fact streamed from the data center, the time saved is 0.)

As each request description describes R_n requests, the time saved for the entire request description is $R_n \times (L_D - L)$.

To compute the total score for the data set, we sum the time saved for individual request descriptions in milliseconds, multiply by 1000 and divide it by the total number of requests in all request descriptions, rounding down.



A schematic representation of the example submission file above.

In the **example** above, there are three request descriptions for the endpoint 0:

- 1500 requests for video 3, streamed from cache 1 with 300ms of latency, saving $1000ms - 300ms = 700ms$ per request
- 500 requests for video 4, streamed from the data center, saving 0ms per request
- 1000 requests for video 1, streamed from cache 2 with 200ms of latency saving 800ms per request

There is also one request description for the endpoint 1:

- 1000 requests for video 0, streamed from the data center, saving 0ms per request

The average time saved is:

$(1500 \cdot 700 + 500 \cdot 0 + 1000 \cdot 800 + 1000 \cdot 0)/(1500 + 500 + 1000 + 1000)$

which equals 462.5ms. Multiplied by 1000, this gives the score of **462 500**.

Note that there are multiple data sets representing separate instances of the problem. The final score for your team will be the sum of your best scores on the individual data sets.