Minimizing Retrieval Latency for Content Cloud

Presenter: Xuanjia Qiu

Oct. 4, 2011

Introduction

- Mathias Bjorkqvist, Lydia Y. Chen, Marko, Vukolic, Xi Zhang, "Minimizing Retrieval Latency for Content Cloud", in Proc. of INFOCOM 2011
- Features:
  - A analytical framework for modelling and analysing the caching policies and source selection strategies in a distributed hybrid content cloud.
  - Hybrid of analytical and numerical methods.
  - Simple and elegant.

Problem

In a hybrid content distribution system consisting a server and a number of geo-distributed edge nodes, how to model the retrieval latency under different caching policies at edge nodes and source selection strategies in a joint framework?

- Users resident in N zones. In each zone, there is an edge node which caches content items and upload content items when a request arrives.
- Edge node: Geo-distributed computers that lie at the edge of the Internet and are able to cache and upload content items upon receipt of requests.
- Retrieval latency: the sum of queueing delay and service time (edge node and the server serve a maximum number of requests simultaneously).
- Source selection: In case of local content unavailability, edge nodes redirect the request to some other edge nodes or the server.

Model

- K items of content are distributed to users when a request for the content is issued.
- A server and N edge nodes.
- The server stores all content. Each edge node can cache B items.
- The server can serve C_s requests simultaneously. Each edge node can serve C_e requests simultaneously.
- Popularity of content k is r_k, which follows Zipf distribution and normalized to fit $\sum_k r_k=1$.
- Request rate is λ for each node.
- Service time: The time needed to upload a content item. Exponential distributions with mean $\frac{1}{\mu^e}$ (at edge node) and $\frac{1}{\mu^s}$ (at server)

Some assumptions easing the modelling

- Homogeneous: edge nodes, request rates, average service time, etc..
- Random process: exponential distribution.
- No retrieval latency for local hit.

Comment at the assumption

It has no property of elasticity which is the fundamental property when we talk about cloud computing platform or SaaS deployed on the cloud platform.

- Upload capacity and cache capacity is bounded by fixed numbers.

Caching policies

- Selfish caching
  - Each edge node cache the B most popular content items. Each node keeps the same set of content items.
- Collective caching
  - Collectively cache the optimal number of copies of content items in the edge network.
  - Number of copies n_k* is proportional to the request rate r_k and
    (1) $0 \leq n_k* \leq N$
    (2) $\sum_k n_k* = NB$.
  - Shortage: the server is not considered, so that the server be underutilized.

Caching policies (cont.)

- Adaptive caching
  - Partition items:
    (1) T_1 *gold* items (most popular, always cache).
    (2) T_2 bronze items (most unpopular, never cache).
    (3) $B - T_1 - T_2$ silver items (less popular, cache when a local miss).
  - Coupled with different *discarding algorithms*:
    (1) Adapt-L (LRU).
    (2) Adapt-R (Randomly).

Source selection

- Random selection: From n_k edge nodes having content k, one is selected with the probability of $\frac{1}{n_k}$.
- Shortest-Queue selection: From n_k nodes having content k, the node which has the shortest request queue is selected.

Framework of the solution

$$D = \Psi^e D^e + \Psi^s D^s$$

- Ψ^l: local hit ratio, Ψ^s: retrieval ratio of the server, Ψ^e: retrieval ratio of the edge network
- D: total latency, D^e: latency at an edge node, D^s: latency at the server.
  - D^s and D^e depend on Ψ^e, Ψ^s but not vice verse.
  - D^s, D^e can be computed by analysing selection policy only.
  - Ψ_s, Ψ_e can be computed by analysing caching policy only.

Server Retrieval Latency, D^s

- $M/M/C_s$ queue.
- $\lambda^s = N\lambda\Psi^s$
- $\rho^s = \frac{\lambda^s}{C_s\mu^s}$
- Stability condition: $\rho^s < 1$
- D^s only depends on C_s, λ, μ^s.
- According to queueing theory, get: $D^s = \frac{Q^s}{\lambda^s}$, where

  $Q^s = C_s\rho^s + \frac{\rho^s}{1-\rho^s}\frac{(C_s\rho^s)^{C_s}}{C_s!(1-\rho^s)} \times \pi_0,$

  $\pi_0 = \{\sum_{k=0}^{C_s-1}\frac{(C_s\rho^s)^k}{k!} + \frac{(C_s\rho^s)^{C_s}}{C_s!}\frac{1}{1-\rho^s}\}^{-1}.$

Edge Node Retrieval Latency, D^e: Random Selection (D^e_{rnd})

- N independent $M/M/C_e$ queues. Further simplified to be $M/M/1$ queue: $C_e = 1$.
- $D^e_{rnd} = \sum_{i=1}^{N} \frac{1/(\mu^e - \lambda^{e_i})}{N}$.
- If all items are uniformly distributed:
  $D^e_{rnd} = \frac{1}{\mu^e - \lambda^e_{rnd}}$
  $\lambda^e_{rnd} = \lambda \Psi^e$.

Edge Node Retrieval Latency, D^e: Shortest-Queue Selection (D^e_{rnd})

- Compare with $M/M/C_eN/JSQ$ queueing model which assumes content items can be retrieved from all C_eN edge nodes. (JSQ = Join Shortest Queue)
- Replace C_eN with C_eN^*
  $N^* = \mathbf{E}_{\mathbf{k}\in\{\mathbf{0}<\mathbf{n_k}<\mathbf{N}\}}[\mathbf{n_k}]$.
  Arrival rate: $\lambda^e_{SQ} = N\lambda\Psi^e$
  Service rate: $C_eN\mu^e$.

Selfish Policy

- From the server: $\Psi^s = \sum_{k=B+1}^{K} r_k$
- From the edge node: $\Psi^e = 0$

Adaptive Caching

- $\Psi^s = \sum_k r_k(1-\pi_k)^N$
- $\Psi^e = \sum_k r_k \psi_k^e = \sum_k r_k(1-\pi_k)(1-(1-\pi_k)^{N-1}$.
- π_k: Steady state probability of item k being cached.

π_k–LRU

- LRU stack
- π_k: steady state probability of content item k being cached in a node.
- $b_k(j)$: probability that content item k is contained in one of the first j positions. $(\pi_k = b_k(B - T_1))$.
- $p_k(j)$: probability that content item k is at location j of the LRU list. $b_k(j) = \sum_{l=1}^{j} p_k(l)$....(1).
- $a_k(j-1)$: conditional probability that content k is moved from location $j-1$ to location j, given that a content item is moved from location $j-1$ to j.
  Approximate: $p_k(j) = a_k(j-1)$.

π_k – LRU (cont.)

- $s_k(j) = r_k(1 - b_k(j-1))$: the rate at which content item k is pushed down from location $j-1$ due to local request (same as rate brought into the top $j-1$ positions)
- $t_k(j) = \psi_k^e(b_k(B - T_1) - b_k(j))$: the rate at which content item k is moved from location j to $j-1$ due to edge request (same as rate brought into the top $j-1$ positions).
- $p_k^{(j)} = a_k(j-1) = \frac{s_k(j)+t_k(j)}{\sum_k(s_k(j)+t_k(j))}$ ...(2)
- Solve (1) (2) iteratively by setting initial value of π_k as r_k.

π_k – Randomly

- $U = \sum_{k \in \{Silver\}} r_k(1 - \pi_k)$...(1): total rate at which a content item is brought into the local buffer.
- Discarding rate: $\frac{\pi_k}{B - T_1}$.
- Conservation law: $r_k(1 - \pi_k) = \frac{\pi_k}{B - T_1} U$.
- $\pi_k = \frac{r_k}{\frac{U}{B - T_1} + r_k}$.... (2)
- Solve (1) (2) iteratively

Summary and discussion

- Numerical results showing that the model can fit the real system very well.
- This paper tells us by combining standard techniques and old results of other researchers in a clever way, one can write a paper accepted by INFOCOM. The key is to make proper approximation.
- Although an analytical result in the form of numerical experiments is not as good as closed-form analytical results, for a difficult problem, effective numerical computation can be satisfactory as well.