Edge Caching

Final Presentation

Problem at hand:

Effect of network densification on edge caching performance.

- Conducted experiments on a simple topology, to verify data given in existing literature.

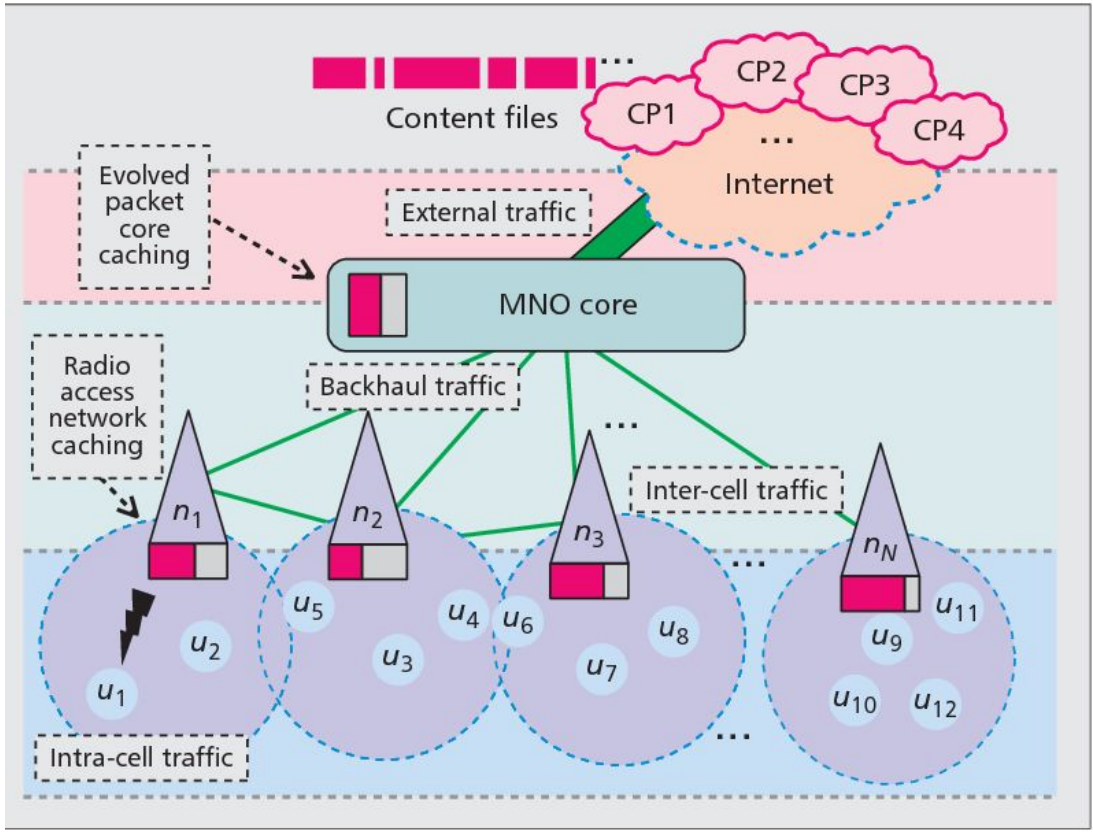
- Had to implement realistic content request patterns.

- Created new topology, dense, and ran experiments on it.

I started off with this.

One Ran per location, with less

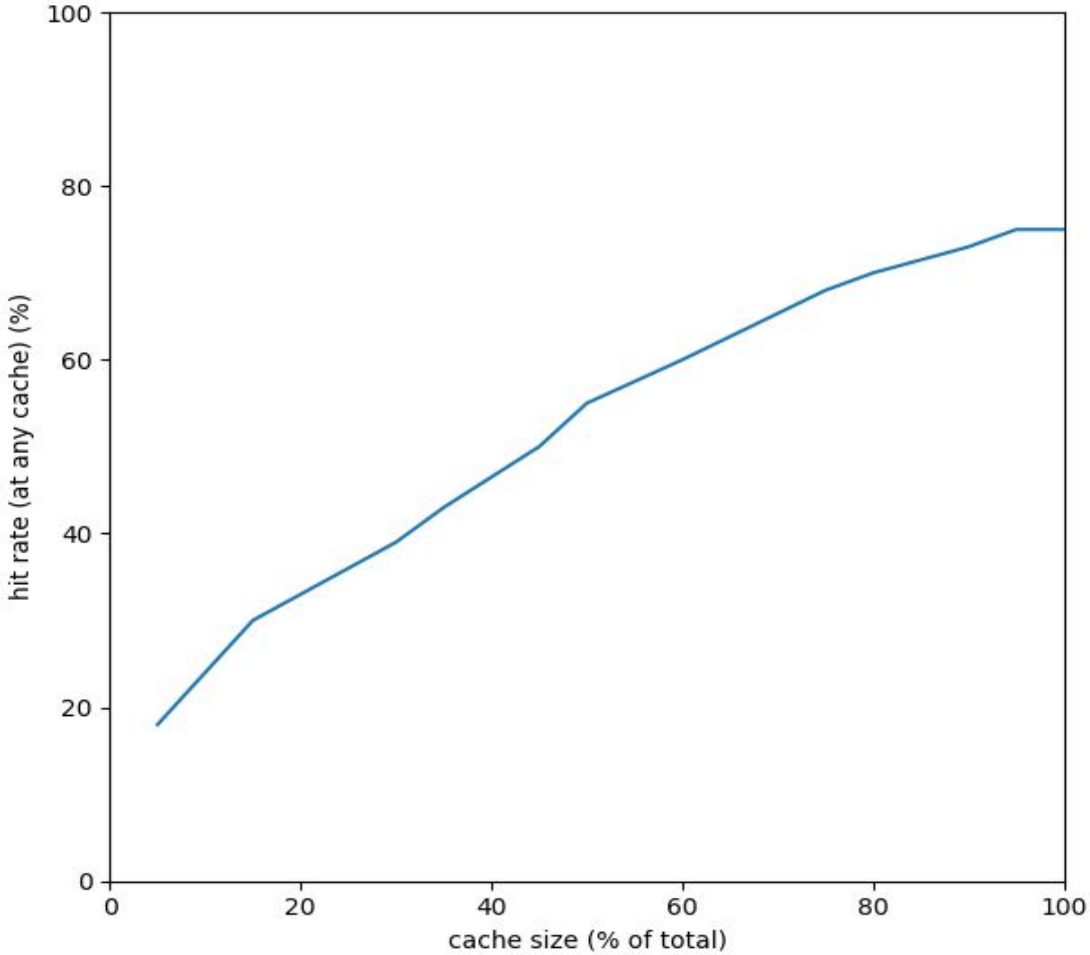
users for each location.



X axis:Total Cache size alloted (% of total content)

Y axis: Hit rate (%)
(at any cache)

The decreasing slope is intuitive.



Realistic request patterns

Referred :

- Performance Study of Load Balancing Algorithms in Distributed Web Server Systems, Zhong Xu, Rong Huang. (2009), [1]
- Impact of Traffic Mix on Caching Performance in a Content-centric Network, Christine Fricker, Philippe Robert, James Roberts, and Nada Sbihi. (2012). [2]
- Generating Web Traffic Based on User Behavior Model.(2013) Guo-feng Zhao, Min-chang, Yu Chuan Xu, Hong Tang. [3]
- Performance and Precision of Web Caching Simulations Including a Random Generator for Zipf Request Pattern(2106), Gerhard Hasslinger,Konstantinos Ntougias, Frank Hasslinger [4]

Realistic request patterns

- [1] showed on average, 75 percent of the client requests come from only 10 percent of the domains. Showed 2 approximations

    - Zipf-like distribution.
      $P(i) = \Omega / i^{\alpha}$, where $\Omega = ({}^{N}\Sigma_{i=1}\ 1/\ i^{\alpha})^{-1}$

    - Geometric distribution.
      $N_i = p(1-p)^{1-i}$, where p is the parameter between [0,1]. (for i^{th} file).

But they proceed with the Zipf-distribution.

Realistic request patterns

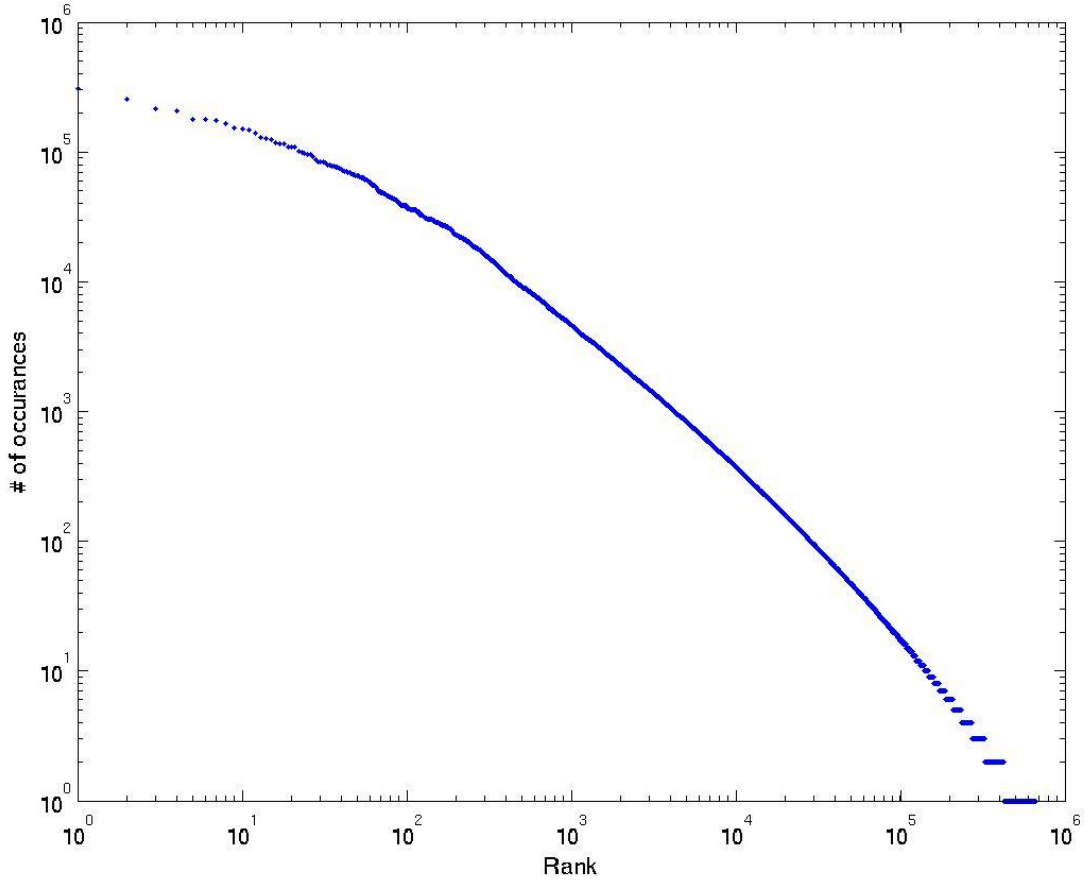
[2] further explores the value and range of α, the parameter which determines the skewness of the zipf-distribution.

$.75 < \alpha < .82$ matched data from many popular web-pages.

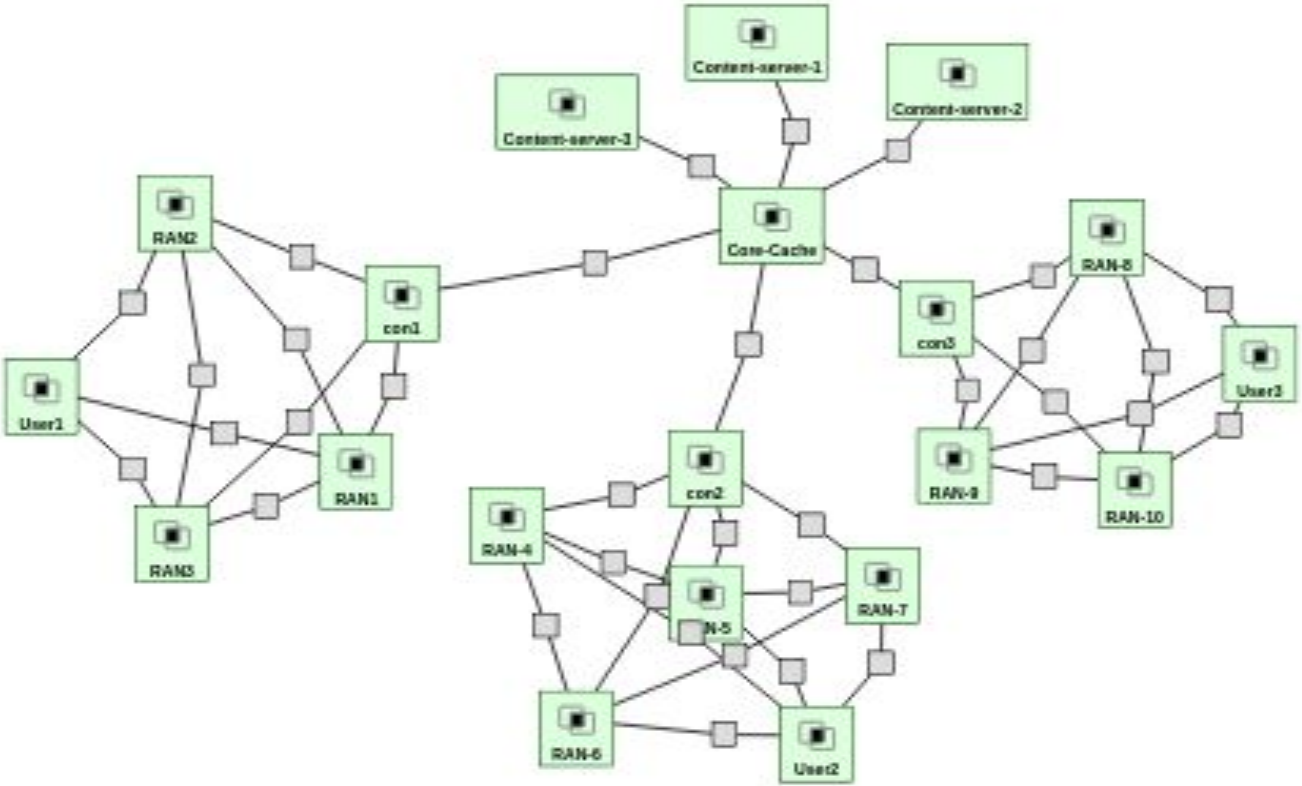
Although, it changed for video content, with the distribution getting more skewed, and α going up to 1. Popular video sites like youtube had an inconsistent distribution, $\alpha = 1.2$ for higher ranked videos, and $\alpha = .5$ for lower ranked videos.

Zipf: α = .80

On a log-log scale



Topology used:



Standard constants for my experiments

- Content server stored files:
  Majority (55%) of them were of size 150 KB to 1.5MB.
  Some (35%) were very small files. < 150 KB
  Very few (10%) weere large files, <5 MB

- Max File size : Small cell: 1 MB
  Core cache: 3 MB

- Capacity of Cache storage: Small Cell: 500 MB
  Core Cache: 2000 MB

Standard constants for my experiments

- 2000 requests per each user.
- Every user followed the same request pattern. : Zipf - like dist., $\alpha = 0.8$
- Pre-cache: 1st 300 ranked files.

No. of users per Small Cell kept changing for each exp, so did the no. of Small cells.

Experiments

I used one user node for all virtual users for each location.

The variables changing were no. of users per SC, and the number of small cells at each location.

I used the simple Cache placement policy - Leave copy down.

Experiments

I first ran my an experiment for 2 different sets of files:

- Ranked randomly.
- Ranked such that low size files are (mostly) higher ranked.

Both the cases gave similar Hit-Rates : 65% - 67%

But Local Hit rates were significantly different: 33% - 45%

Proceeded with the 2nd case for rest of experiments.

Experiments

At first: 2 locations with 3 Small Cells, and one location with 4 Small Cells.

I went from 1 user per SC to 5:

- Overall Hit rate: 65%, Core Hit rate: 38.6%, local hit rate: 43% - 42%
- Overall Hit rate: 63%, Core Hit rate: 40.1%, local hit rate: 41% - 42%
- Overall Hit rate: 61%, Core Hit rate: 42.2%, local hit rate: 36% - 43%
- Overall Hit rate: 59%, Core Hit rate: 43.5%, local hit rate: 32% - 36%
- Overall Hit rate: 55%, Core Hit rate: 44.8%, local hit rate: 29% - 33%

Experiments

Then I varied the number of Small Cells at each location, kept no. of users 10 at each location: went from 1 small cells to 4 at each location.

- Overall Hit rate: 65%, Core Hit rate: 35.5%, local hit rate: 60%
- Overall Hit rate: 63%, Core Hit rate: 38.6%, local hit rate: 50.2%
- Overall Hit rate: 59%, Core Hit rate: 40.6%, local hit rate: 42.8%
- Overall Hit rate: 56.8%, Core Hit rate: 42%, local hit rate: 38.3%

Experiments

I then tried a direct cache system, i.e, a direct connection between User end device and Core-cache. I varied the number of users at each location, and noted the Hit rate at Core-cache:

- Users: 6 - Hit rate: 67.2%
- Users: 7 - Hit rate: 66.9%
- Users: 8 - Hit rate: 67.5%
- Users: 9 - Hit rate: 65.4%
- Users: 10- Hit rate: 64.3%

Limitations

- Couldn’t implement predictive caching
- Space constraints.
- I wasn’t able to use the sibling directive
- Only tested for Web-page content, not Video content which amounts to a large part of today’s web-traffic.

Thank You