Content Store Management in Name Data Networking

Sanchita Badkas
Computer Science and Engineering
Bits, pilani
Pilani, India
h20190520@pilani.bits-pilani.ac.in

Yashita Goswami
Computer Science and Engineering
Bits, pilani
Pilani, India
h20190521@pilani.bits-pilani.ac.in

***Abstract*—Information centric networks have gained a lot of attention in the future internet architecture research due to hard to meet demands of security, mobility and content distribution. This gave rise to Named Data Networks(NDN). In-network caching is the most attractive feature of NDN. This helps in reducing data access latency and mitigate the affects of server bottleneck. In order to enhance the performance further a lot of strategies have come up for caching. We attempt to incorporate machine learning methodology in order to come up with effective cache size.**

***Index Terms*—ICN, NDN, Cache, Optimization, ML**

I. Introduction

Information-centric networking (ICN) is an approach to evolve the Internet infrastructure away from a host-centric paradigm, based on perpetual connectivity and the end-to-end principle, to a network architecture in which the focal point is identified information (or content or data). In this paradigm, connectivity may well be intermittent, end-host and in-network storage can be capitalized upon transparently, as bits in the network and on data storage devices have exactly the same value, mobility and multi access are the norm and anycast, multicast, and broadcast are natively supported.[3] Data becomes independent from location, application, storage, and means of transportation, enabling in-network caching and replication.[3] NDN is a ICN and is similar to CCN (Content centric networks). NDN mainly consists of three components,

A. PIT - Pending Interest Table

The PIT stores all the Interests that a router has forwarded but not satised yet. Each PIT entry records the data name carried in the Internet, together with its incoming and outgoing interface(s).[7]

B. FIB - Forwarding Information Base

FIB is populated by a name-prex based routing protocol, and can have multiple output interfaces for each prex.[7]

C. CS - Content Store

The Content Store is a temporary cache of Data packets the router has received.[7] Automatic in-network caching is enabled by naming data. Since each NDN Data packet is meaningful independent of where it comes from or where it may be forwarded to, a router can cache it in its content store to satisfy future requests. Upon receiving a new Interest, the router first checks the Content Store. If there is a data whose name falls under the Interest's name, the data will be sent back as a response. [8] Hence, the kind of strategy we use to implement the content store plays a role in reducing the latency to fetch the data upon request. We intend to explore these strategies in order to find the most efficient one.

II. Goals

We attempt to solve one of the key challenges in CCN deployment that is the cache allocation problem: given a finite set of cache storage and a specific topology, how should the storage be distributed across the CCN routers?[2] We know that, after a point the size of cache adds from little to no value to the efficiency of NDN. We will consider hit ratio as our metric for estimating efficiency. Based on the various traffic patterns we aim to reduce the cost of caches by giving out the optimal cache size as, a CCN router with 10 TB of cache space using Flash-based solid-state drives (SSDs)would cost $300;000 and consume 500 W of power[6] We aim to find patterns in required cache size based on certain set of parameters for a small topology and check if the similar patterns apply in the scaled up topologies.

III. Assumptions and simulation environment

A. Assumptions

- We are assuming a dynamic cache which would act like a plug-n-play device. Thus, whenever extra cache size is not required we can remove it. In case, the size allocated is less we can simply plug in the extra cache.
- Since most of the traffic patterns follow ZIPF distribution with 0.7 skewness, we will be using the same in our configurations. [1][2]
- The NDN design assumes hierarchically structured names, hence we are assuming a binary tree for our topology.
- Also, as it is a hierarchical structure, it's assumed that only root is the producer and leaf nodes are consumers while the internal nodes are the routers.

B. Simulation environment

We will be fixing a certain set of parameters like cache replacement policy used is Least Recently Used(LRU), packet size is 1024 bytes and the routing strategy used is best path strategy.

IV. Implementation details and architecture

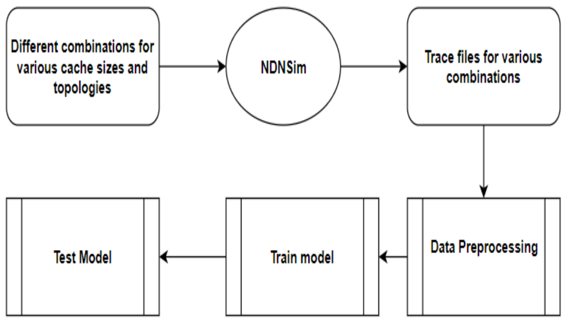
Different combinations for various cache sizes and topologies
NDNSim
Trace files for various combinations
Data Preprocessing
Train model
Test Model

Fig. 1. Steps involved in predicting the cache size

A. Different combinations for various cache sizes and topologies

We created some topology files for level 3 and level 4 complete binary trees. Interest rates considered for simulations are 300,500,700 and 1000 interests per sec. For all these combinations, we tried out different cache sizes while maintaining the rule that,
Cache size of root $>$ Cache size of L1 $>$ Cache size of L2 $> \ldots . >$ Cache size of leaf nodes

B. NDNSim

All the consumer node applications start after 0.1 secs one after the another and send requests to the root node at the specified interest rate.

C. Trace files for various combinations

Using CSTrace function of NDNSim, we extracted the CS trace for all the simulations. CS trace gives the hit and miss statistics of all the nodes at a particular instance.

D. Data preprocessing

The statistics from the CS trace file were converted to get hit ratio for each node in the topology. The hit ratio, along with other parameters like interest rate, cache size, distance from producer, distance from consumer, total levels, level the node is located at were collected together and stored in a CSV file for further analysis various graphs were plotted for feature extraction. Based on the insights of the graphs, interest rate was dropped from the features to be considered for the model. Since, hit ratio is indirectly dependent on the cache size, it is removed to avoid data leakage. Outlier data where the hit ratio was constant for the cache size was removed in order to make it easier for the machine to detect patterns.

E. Train

The processed data was used to train the model. We have used random forest regressor for this purpose.

F. Test

The machine was trained using the trace files of level 3 and level 4 complete binary trees. In order to test, whether the model is able to output optimized cache size for scaled up topologies we used the trained model to predict the values for level 7 and level 9 topologies. The output was compared with various other cache combinations.

V. Observations

For level 3 and Level 4 topologies, we made the following observations using Fig 2 and Fig 3,

- Interest rate doesn't affect the cache size for the assumed configurations.
- Hit ratio for root doesn't go above 0.5 for most of the cases. In handful of cases where it does go above it, the hit ratio for lower levels drop.
- For all levels other than the one closest to the producers and consumers, the hit ratio drops down to 0.5 after a certain amount of cache size.
- On the contrary, the levels closest to the consumers give about 96 % accuracy for all cache sizes after a certain size.
- Most levels have hit ratio of at least 50% for the assumed configurations.

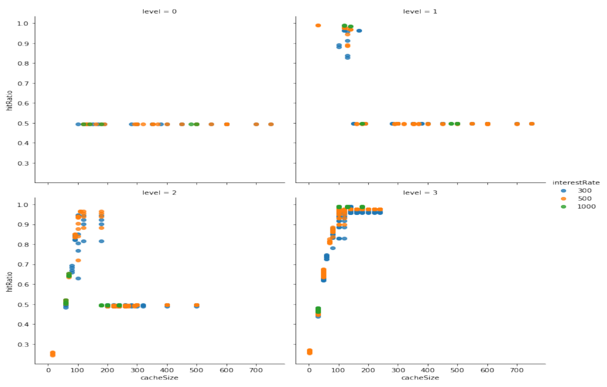

Fig. 2. Level 4 complete binary tree topology

In order to test how the behaviour would change if we consider ternary tree of level 3, we tried out different combinations for ternary topology and made the following observations,

- As observed previously, hit ratio of routers near the consumers stays above 90% after a certain cache size.
- Hit ratio for root was raised to 66% from 50% which was the case in binary tree topology

VI. Results

The model gave an accuracy of 63.22%, when tested using train-test-split method for level 3 and level 4 topology. We tested the trained model for level 7 and level 9, we observe

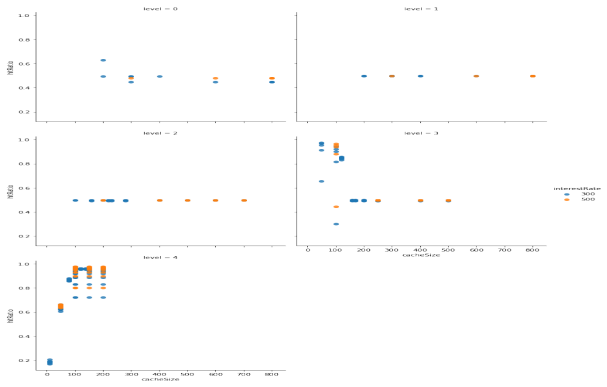
Fig. 3. Level 5 complete binary tree topology

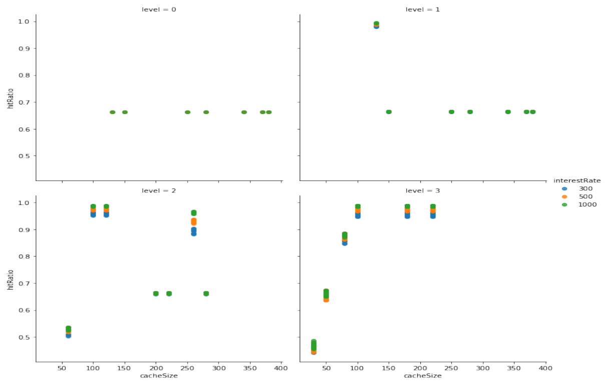
Fig. 4. Level 3 complete ternary tree topology

from Fig 5 and Fig 6 that we get a rough estimate of the cache size required but it isn't optimal.

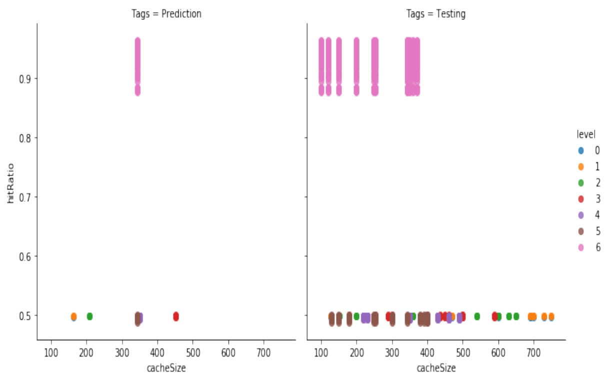

Fig. 5. Predictions vs Random in Level 7 tree

VII. Can the score be improved?

There are a lot of assumptions being made about the traffic flow, topology and other configurations for this model. If we provide the data to the model for these combinations, the number of dimensions given for training would increase leading to more insights into the cache allocation pattern which might improve the results of our system and would be applicable for a wide range of topologies and configurations. Also, the amount of data given to the machine for the assumed topology was very noisy even after data preprocessing as it was obtained using a hit and try strategy. Due to this, there is a high possibility that the model suffered from overfitting leading to less accuracy. If we could obtain more dimensions and less noisy data, the accuracy might improve.

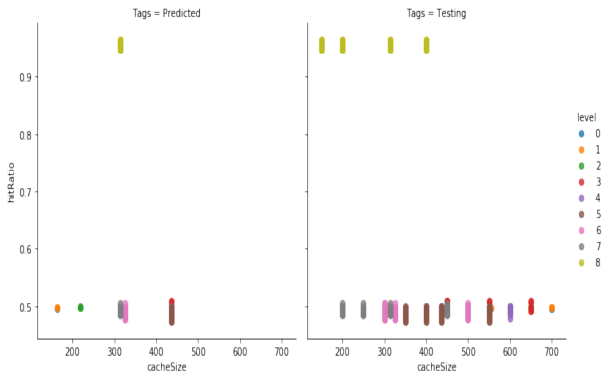

Fig. 6. Predictions vs Random in Level 9 tree

VIII. Conclusion

As from the above observations, we conclude that using machine learning and right data we can develop a model that can predict optimal cache size which would in turn save the cost.

References

[1] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and Zipf-like distributions: Evidence and implications," IEEE INFOCOM, 1999
[2] Yonggong Wang, Zhenyu Li, Gareth Tyson, Steve Uhlig, and Gaogang Xie, "Design and Evaluation of the Optimal Cache Allocation for Content-Centric Networking",IEEE TRANSACTIONS ON COMPUTERS, VOL. 65, NO. 1, JANUARY 2016
[3] https://en.wikipedia.org/wiki/Information-centric-networking
[4] Q. Wu, Z. Li, J. Zhou, H. Jiang, Z. Hu, Y. Liu, and G. Xie, "Soa: Toward service-oriented information centric networking," IEEE Netw., vol. 28, no. 3, pp. 12–18, May 2014.
[5] G. Tyson, S. Kaune, S. Miles, Y. El-khatib, A. Mauthe, and A. Taweel, "A trace-driven analysis of caching in content-centric networks," in Proc. IEEE Int. Conf. Comput. Commun. Netw., 2012, pp. 1–7.
[6] D. Perino and M. Varvello, "A reality check for content centric networking," in Proc. ACM SIGCOMM Workshop Inf.-Centric Netw., 2011, pp. 44–49.
[7] Named Data Networking by Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang ACM SIGCOMM Computer Communication Review (CCR), July 2014.
[8] https://named-data.net/project/archoverview/