Project Proposal

Collaborative Caching With Interesting Eviction Algorithms on Multiple Datasets

Team Members
Akash (823008707)
Anish Dubey (523002969)
Nishant Gill (924001574)
Nikhilesh Pandey(224002412)

Course
CSCE 613: Advanced Operating Systems

Advisor
Dr. Daniel A. Jimenez

Contents

1. **Basic Idea**
2. **Collaborative Cache**
3. **Consistent Hashing**
4. **Eviction Algorithms**
5. **Cache Size**
6. **Cache Literature Review**
7. **Goal**
8. **Preliminary Results**
9. **Data Sets**
10. **Tools and Methodology**
11. **Work Distribution Among Members**
12. **References**

1. Basic Idea

Our basic idea is to build a collaborative cache using consistent hashing with various eviction algorithms like FIFO, LFU, LRU, SxLRU etc. Collaborative Cache has already been implemented and suggested in the relevant architecture [1]. Our idea is to implement the collaborative caching with various eviction algorithms on various datasets.

2. Collaborative Cache

This cache technique will be used on different datasets and performance will be measured against independent cache implementations with no collaboration. The idea behind collaborative cache is that it combines multiple caches as a single cache of bigger size and has an added advantage of being able to find a document with high probability but with increased latency. Latency is an important factor to consider in case of collaborative cache but just like the paper assumes that the advantage of implementing collaborative cache outweighs than latency, we will also implement the cache without considering latency factor.

3. Consistent Hashing

Collaborative caching is one such mechanism in which the cache available on distributed servers or storage devices form a single cache to serve the requests. There are different ways to implement hashing for the cache implementation. We plan to use consistent hashing for selecting different caches. Consistent hashing helps to find a cache in a distributed cache system to store or retrieve data from cache, while at the same time being able to cope with cache additions and failures at run time. As a result, consistent hashing helps in scaling the collaborative cache without outside human intervention.

4. Eviction Algorithms

We will be implementing various evictions algorithms on our collaborative caching like FIFO, LRU , LFU etc. One important eviction algorithm discussed in the paper [] was S4LRU. We will also implement the segmented LRU and will investigate the idea that whether the 4 is a magic number here or it depends on the dataset given. Hence, our motive will be to find out the optimum value of x in case if SxLRU outperforms the other evictions algorithms. This value may differ for different datasets.

5. Size

We will also see the effect of varying the size of the cache against the hit ratio. Intuitively, the caching performance should improve with increase in the size of the cache. We will test whether this is true in our data sets too or not.

6. Cache Literature Review

We will also look into various caching algorithms discussed in context of various computer systems areas like operating systems, distributed systems etc. We will see that depending on the data type what other caching algorithms will be useful for our datasets.

7. Goal

- We plan to simulate the collaborative cache implementation on various data sets mentioned below using consistent hashing.
- Compare the improvement in data access from cache in both independent and collaborative cache implementations on the data sets.
- Implement various eviction algorithms for the collaborative cache in order to compare the efficiency of caching algorithms.
- Implement the SxLRU proposed in the paper 'Analysis of the facebook photo caching' and find the parameter x for the various data sets.

8. Preliminary Results

We have taken "An Analysis of Facebook Photo Caching " paper as a base paper. The paper has used collaborative caching using S4LRU and have seen 17% improvement on cache hit ratio. The value 4 chosen in S4LRU turns out to be specific to a related dataset. It will be interesting to see what value of x in SxLRU gives us the highest cache hit ratio.

9. Data Sets

(i) Web Traffic (Indiana University)

A collection of web (HTTP) requests for the month of November 2009.
JSON object format for the dataset:

```
{
  'timestamp': 123456789, # Unix timestamp
  'from': '...', # the referrer host
  'to': '...', # the target host
  'count': 1234 # the number of request between the referrer and target hosts that occurred
within the given hour
}
```

Dataset statistics

- Dataset size: 235 mil requests
- File size: 2.7GB uncompressed
- Time period: Nov 1, 2009 – Nov 22, 2009

(ii) Twitter Dataset
A collection of records extracted from tweets for the month of November 2012 containing both #hashtags and URLs as part of the tweet.
JSON object format:

```
{
  'timestamp': 123456789, # Unix timestamp
  'user_id': 12345, # an integer uniquely identifying the user who tweeted
  'hashtags': ['...', '...', '...'], # a list of hashtags used in the tweet
  'urls': ['...', '...', '...'] # a list of links used in the tweet
}
```

Dataset statistics

- Dataset size: 22.5 million tweets
- File size: 2.5GB uncompressed
- Time Period: Nov 1, 2012 – Nov 30, 2012

10. Tools and methodology:
We will use JAVA and JDK for simulating the environment and implementing the various algorithms. We might also use Hadoop distributed file system (HDFS) and map-reduce framework to process the large data-sets if needed.

11. Work Distribution among members:
Team will split the work on implementing various algorithms and test data sets. We have two different data sets which team members can split among them. Two members will work on each data set and similarly on different algorithms.

12. Reference
[1] http://www.cs.cornell.edu/~qhuang/papers/sosp_fbanalysis.pdf

[2] http://www.cs.rochester.edu/~cding/Documents/Publications/ismm11a.pdf

[3]https://www.ipvs.uni-stuttgart.de/export/sites/default/ipvs/abteilungen/as/lehre/lehrveranstaltungen/vorlesungen/WS1415/material/ARC.pdf

[4] http://dl.acm.org/citation.cfm?id=2487788.2488174 (for dataset)
[5] http://informatics.indiana.edu/fil/Papers/click.pdf (for dataset)