A summary on

Chapter 13: Measuring Efficiency

By Akshat Kukreti

Throughput and Latency

- Throughput: Number of queries a search engine processes within a given period of time.
  - Measured in queries per second (qps).
  - *Theoretical throughput* or *service rate*
  - Service time: The amount of time per query that the processor(s) is/are actively working on the query.
  - Service rate = no. of processors/Service time

Throughput and Latency

- Latency: Time elapsed from the search engine's receipt of the query until the results are sent to the user.
  - Measured in seconds per query.
  - *End-to-end latency:* Indicator of user satisfaction
- Latency and throughput aggregated for large number of frequencies and reported as *mean throughput* or *mean latency*.

Throughput and Latency

- Latency > service time
- Splitting query processing load
- Index replication: Increases throughput but no effect on latency.
- Index partitioning: Increases both throughput and latency by a similar degree.

Caching

- Reducing average service time by caching frequently used pieces of data.
- Three-Level Caching
- Query frequencies follow Zipfian distribution i.e. a long tail of singletons
- 44% of the query volume is caused by singletons
- A cache hit rate lower than 50% are not unusual.

Cache Hierarchy

- Long and Suel proposed a cache hierarchy that interacts with the search engine on three different levels:
- Search results
- Lists intersections – conjunctive retrieval model
- Postings lists – Keeping frequently accessed postings lists in the memory.
- Documents

Cache policies

- Static: The set of cached items is determined ahead of time and is kept fixed.
- Maybe updated periodically, at the time of index refresh.
- Dynamic: The set of cached items may change when a new object is encountered. Also called *replacement algorithms*.

Cache policies

- General purpose cache policies:
  - Least Recently Used (LRU): The item in the cache that has not been accessed for longest time is evicted.
  - Least Frequently Used (LFU): The item in the cache that has been accessed the least frequently (since it was loaded to the cache) is evicted.
- LRU and LFU work well on when operating on a fixed sized set of objects.

Cache policies

- Cost-aware cache policies
  - Split the cache into 3 buckets, one for each caching level.
  - Each cacheable item has the following attributes
    - Cost **c** (in bytes)
    - Gain **g** (Example Service time of query, time required to load a postings list to the memory)
    - No. of times the item is expected to be used **p**
- Expected Net Benefit (ENB) = p*g/c
- Sort all items by their ENB values and store the top k items in the cache.

Pre-fetching search results

- If we have a query log, we can use it to pre-fetch results for queries that we expect to be asked in the future. Example News
- Assuming that the queries for tomorrow will be similar to the queries for today, a set of popular queries can be issued and the results can be cached.
- Continuous pre-fetching