Integral Caching using Online Mirror Descent in a Networked Context

QUENTIN OSCHATZ

The Problem

Caches are used to serve file requests that would otherwise need to be sent to a remote server. These caches cannot store all data that could be requested, and file popularity may change over time. As such, the policies must adapt together with the data, learning in an online manner.

Caches can store data in a fractional or in an integral manner. The former allows arbitrarily small chunks of files to be stored, the latter does not. In [1], the authors show that, in practice, most caching situations can be modeled in an integral manner.

When expanding the problem to multi-cache systems, decisions must now also take into account which files are already stored in other caches, as well as the network topology.

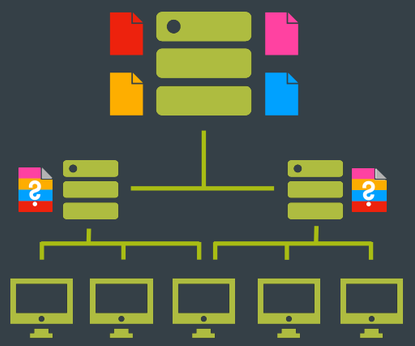

Figure 1: A model of a bipartite caching network. A caching policy must decide which files to cache in the middle two caches in order to maximize the number of requests served by them

Existing Algorithms

Offline Learning
Use ML on historical data to generate a static cache state. Relies on future requests following historical trends.

Least Frequently/Recently Used (LFU/LRU)
Keep the most recently/frequently used files, always add the last requested file. Results in inconsistent performance across different request patterns.

Online Gradient Descent (OGD)
Use gradient descent on a cost function to move towards an optimal state [2]. Can be expanded to bipartite networks [2]. Can only handle one request at a time, and uses additive update functions [3].

Online Mirror Descent (OMD)
Generalized OGD using mirror descent [3]. Allows multiplicative updates and request batching, decreasing performance overhead. Currently, the algorithm is designed for only single cache systems.

Regret

Regret is a metric that can be applied to caching systems in order to examine their performance in an adversarial setting [2]. There, requests are not sampled from a distribution, but deliberately chosen to cause cache misses [2]. Ideally, regret should grow sublinearly compared to time, approaching 0 for high values of t. It has been shown that no deterministic integral policy has a regret guarantee, meaning that any integral policy without online learning (i.e. LRU , LFU) can be "beaten" by an adversary [2].

$$\text{Regret}_T(\delta) = \sup_{\{x_1,\ldots,x_t\}\in\mathcal{X}^T} \left\{ \sum_{t=1}^{T} C_{x_t}(y_t(\delta)) - \sum_{t=1}^{T} C_{x_t}(y^*) \right\}$$

Bipartite OMD

Single cache: $U_{x_t}(y_t) = \sum_{n\in\mathcal{N}} x_{t,n} \cdot (C(0,n) - (y_{t,n} \cdot w_n))$

Network: $U_{x_t}(y_t) = \sum_{n=1}^{N} \sum_{i\in\mathcal{I}} x_{t,i,n}(C(0,i,n) - C(v_t,i,n))$

$$C(v_t,i,n) = \sum_{j\in\mathcal{J}} (w_{i,j,n} - w_{i,j-1,n}) \left(1 - \min\left\{1, \sum_{k=1}^{j-1} v_{t,k,n}\right\}\right)$$

The new utility function checks if the requested file is already cached in a connected cache, and does not reward these redundant files.

OMD is not integral by default, but the online coupled rounding scheme proposed in [3] can create integral states preserving regret bounds.

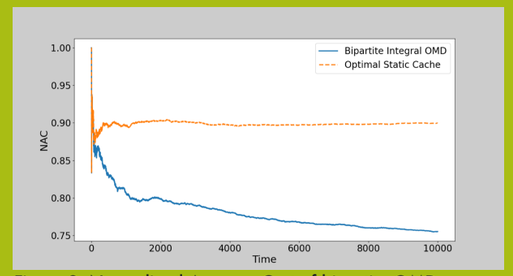

Figure 2: Normalized Average Cost of bipartite OMD vs optimal static in hindsight on a fixed-popularity trace.

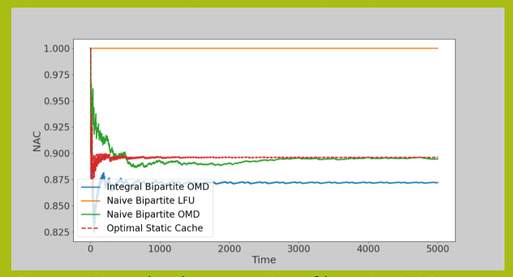

Figure 3: Normalized Average Cost of bipartite OMD vs naive, non-networked OMD and LFU and the optimal static policy, on a circular adversarial trace.

Conclusion

Testing in a bipartite network with three caches and two sources, with one cache connected to both sources and the other two connected only to one source, initial results are very promising. Bipartite OMD has a decreasing Normalized Average Cost, and outperforms naive, non-networked implementations. This implies sublinear regret on this trace, though no formal bounds have been proven.

The algorithm shows promise, bringing the advantages of OMD to the bipartite setting.

Future Work

Future work could include testing hierarchical topologies, as well as proving theoretical regret bounds for this algorithm. Allowing for the use of adoptive learning rates may also prove useful, as different parts of the network serve different users that may require different adaptability. Additionally, in order to improve performance, the more efficient projection algorithm developed in [3] could be extended to the bipartite setting.

TU Delft

[1]L. Wang, S. Bayhan, and J. Kangasharju, "Optimal chunking and partial caching in information-centric networks," *Computer Communications*, vol. 61, pp. 48–57, May 2015, ISSN: 01403664.
[2] G. S. Paschos, A. Destounis, L. Vigneri and G. Iosifidis, "Learning to Cache With No Regrets," *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, 2019, pp. 235-243, doi: 10.1109/INFOCOM.2019.8737446.
[3] T. Si Salem, G. Neglia and S. Ioannidis, "No-Regret Caching via Online Mirror Descent," *ICC 2021 - IEEE International Conference on Communications*, 2021, pp. 1-6, doi: 10.1109/ICC42927.2021.9500487.

Notation

- w: Weight matrix
- y: Cache state
- v: Integral cache state
- t: time
- $\mathcal{J}$: Set of caches
- $\mathcal{N}$: Catalog
- $\mathcal{I}$: Set of sources
- y^*: Optimal cache state
- $C(0,n)/C(0,i,n)$: Cost of cache miss
- $U_{x_t}(y_t)$: Utility
- $C_{x_t}(y_t)$: Cost
- δ: Caching policy

q.oschatz@gmail.com
quentinoschatz.com