On Allocating Cache Resources to Content Providers

Weibo Chu, Mostafa Dehghan, Don Towsley, Zhi-Li Zhang

wbchu@nwpu.edu.cn

Northwestern Polytechnical University

Why Resource Allocation in ICN?

Resource allocation is important for networks:

- QoS: delay, throughput, jitters, etc
- DiffServ: video, emails, instant messaging, etc
- Fairness among users, applications
- Market and economics

The above goals are challenging to realize in ICNs due to in-network caching

- Content can be from anywhere in the network, no connection
- Traditional policies (i.e., LRU, RND, FIFO) treat content of different providers in a tightly coupled manner

Problem Introduction

We consider allocating resources of an edge cache among different contending content providers

- A cache shared by users of K content providers.
- Users access files of CPs through the cache.
- Question: how the cache provider (SP) allocate its resources to realize management purposes?

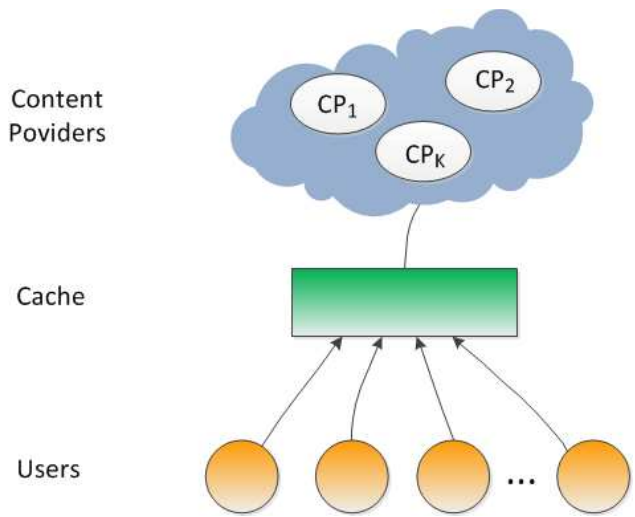


Figure: Network Model.

Problem Formulation

We propose a **cache partitioning** approach.

- The cache provider partitions its cache resource into slices with each slice allocated to one CP;
- Advantage: 1) restricts contention of each CP into its dedicated slice, and hence provides a natural means to tune the performance of each CP; 2) potentially improves system performance; 3) easy-to-implement;

Model Assumptions:

- Cache of size C with LRU policy.
- Each CP k serves N_k different files $F_k = \{f_{1k}, f_{2k}, \ldots, f_{kN_k}\}$ of equal sizes.
- Requests for f_{ik} arrive as a Poisson process with rate λ_{ik}.

Utility-based Cache Resource Allocation

- $U_k(h_k)$: concave, increasing in the hit rate h_k of CP k.
- Partition the cache into K slices, and allocate one with size C_k to CP k.
- Our goal: to maximize the (weighted) utilities over all CPs.

$$\begin{aligned} \text{maximize} \quad & \sum_{k=1}^{K} w_k U_k\Big(h_k(C_k)\Big) \\ \text{such that} \quad & \sum_{k=1}^{K} C_k \le C \\ & C_k \ge 0, \quad k = 1, 2, \ldots, K \end{aligned} \tag{1}$$

Cache Characteristic Time (H.Che 2001)

- For each CP k accessing LRU cache with size C_k, the hit rate h_k can be approximated as:

$$h_k = \sum_{i=1}^{N_k} \lambda_{ik}(1 - e^{-\lambda_{ik} T_k}),$$

where T_k is a constant denoting *cache characteristic time* that satisfies:

$$\sum_{i=1}^{N_k} (1 - e^{-\lambda_{ik} T_k}) = C_k.$$

Important Properties

Theorem 1: Partitioning the cache provides performance gain as compared to sharing it.

$$C = \sum_{k=1}^{K} \sum_{i \in F_k} (1 - e^{-\lambda_{ik} T})$$

It can be seen creating partitions $C_k = \sum_{i \in F_k} (1 - e^{-\lambda_{ik} T})$ provides the same performance.

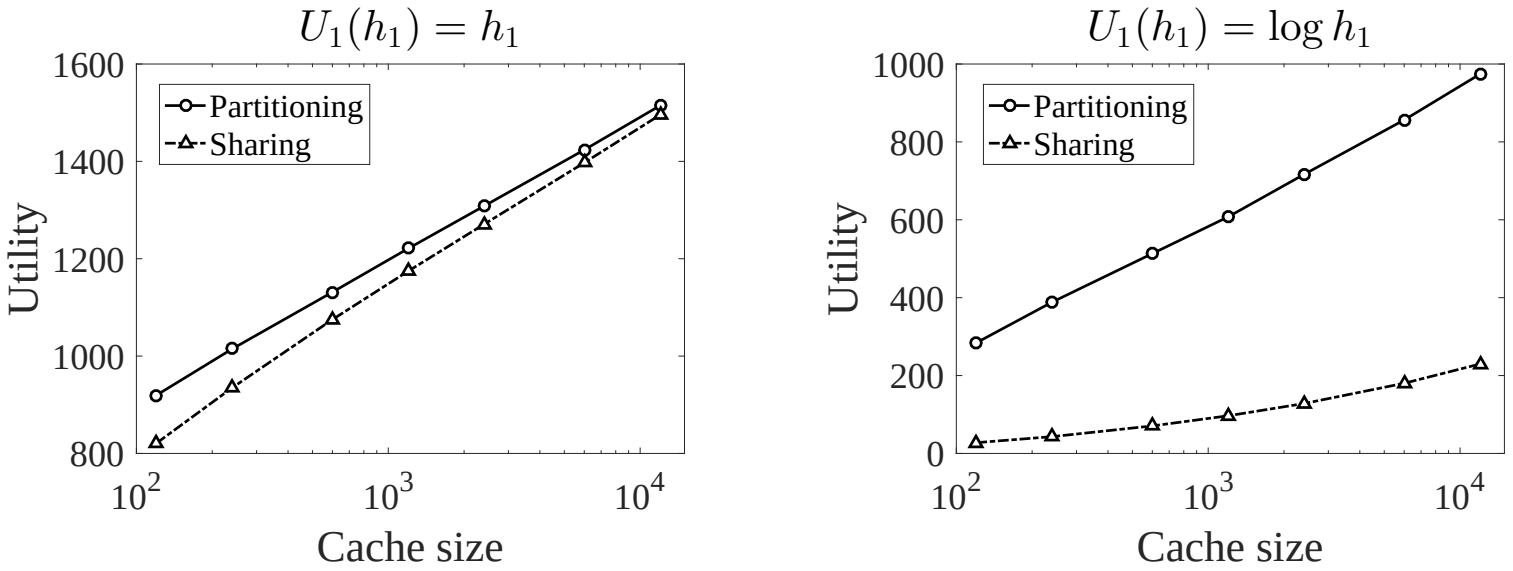
Theorem 2: h_k is concave in C_k, and problem (1) is convex.

Some Numerical Results

Basic setting:

- 2 CPs competing for 1 Cache
- File population: $N_1 = 2 \times 10^5$, $N_2 = 1 \times 10^6$
- Zipf distribution: $\alpha_1 = 1.2$ and $\alpha_2 = 0.8$
- Request rate: $\lambda_1 = 1500$, $\lambda_2 = 1000$
- Utility function: $U_1(h) = U_2(h) = h$.
- Weights: $w_1 = w_2 = 1$

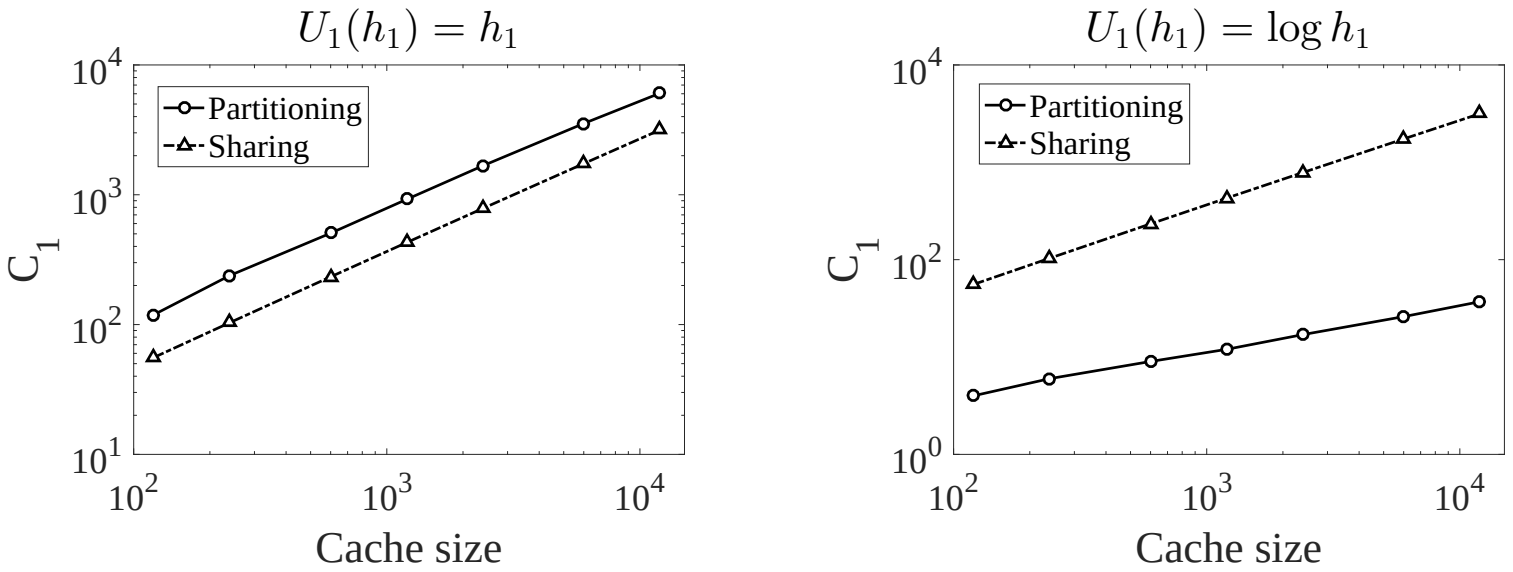
Efficacy of Cache Partitioning



(a) $U_1(h_1) = h_1$ (b) $U_1(h_1) = \log h_1$

Figure: Showing the efficacy of cache partitioning. Aggregate utility obtained by partitioning versus sharing. In both cases, $U_2(h_2) = h_2$.

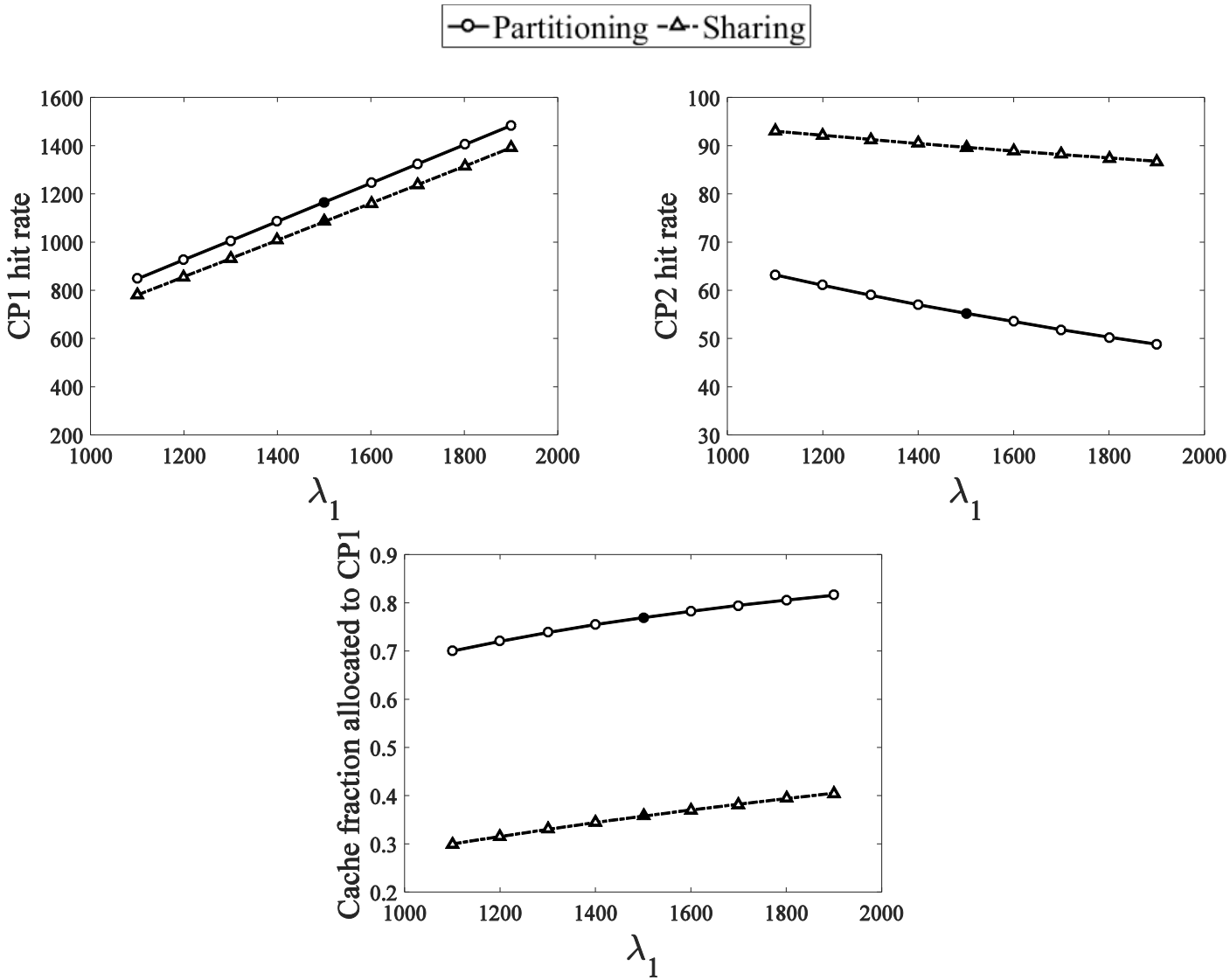
Efficacy of Cache Partitioning



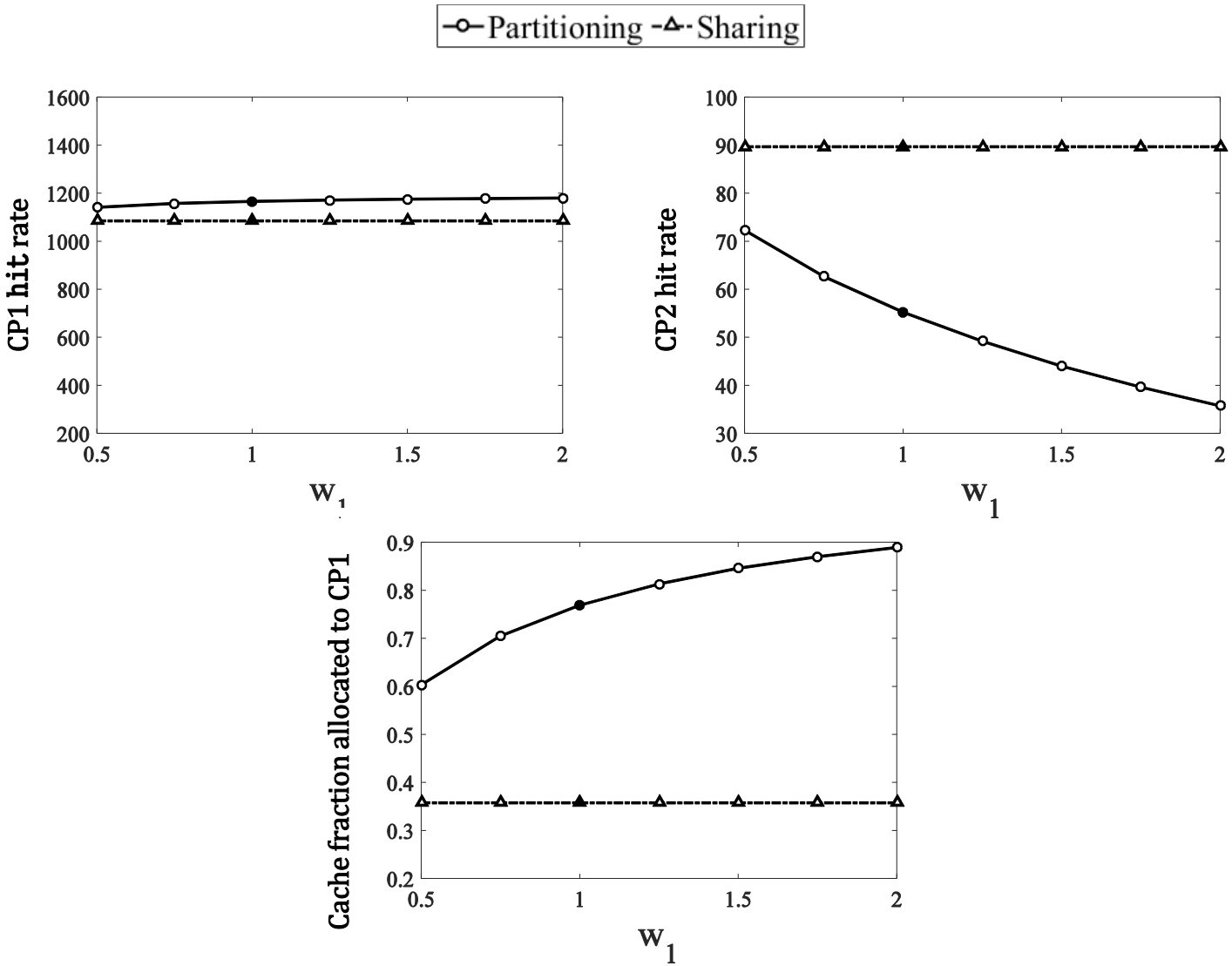
(a) $U_1(h_1) = h_1$ (b) $U_1(h_1) = \log h_1$

Figure: Cache size allocated to CP1 when partitioning the cache compared to average cache storage consumed by CP1 files when sharing the cache. In both cases, $U_2(h_2) = h_2$.

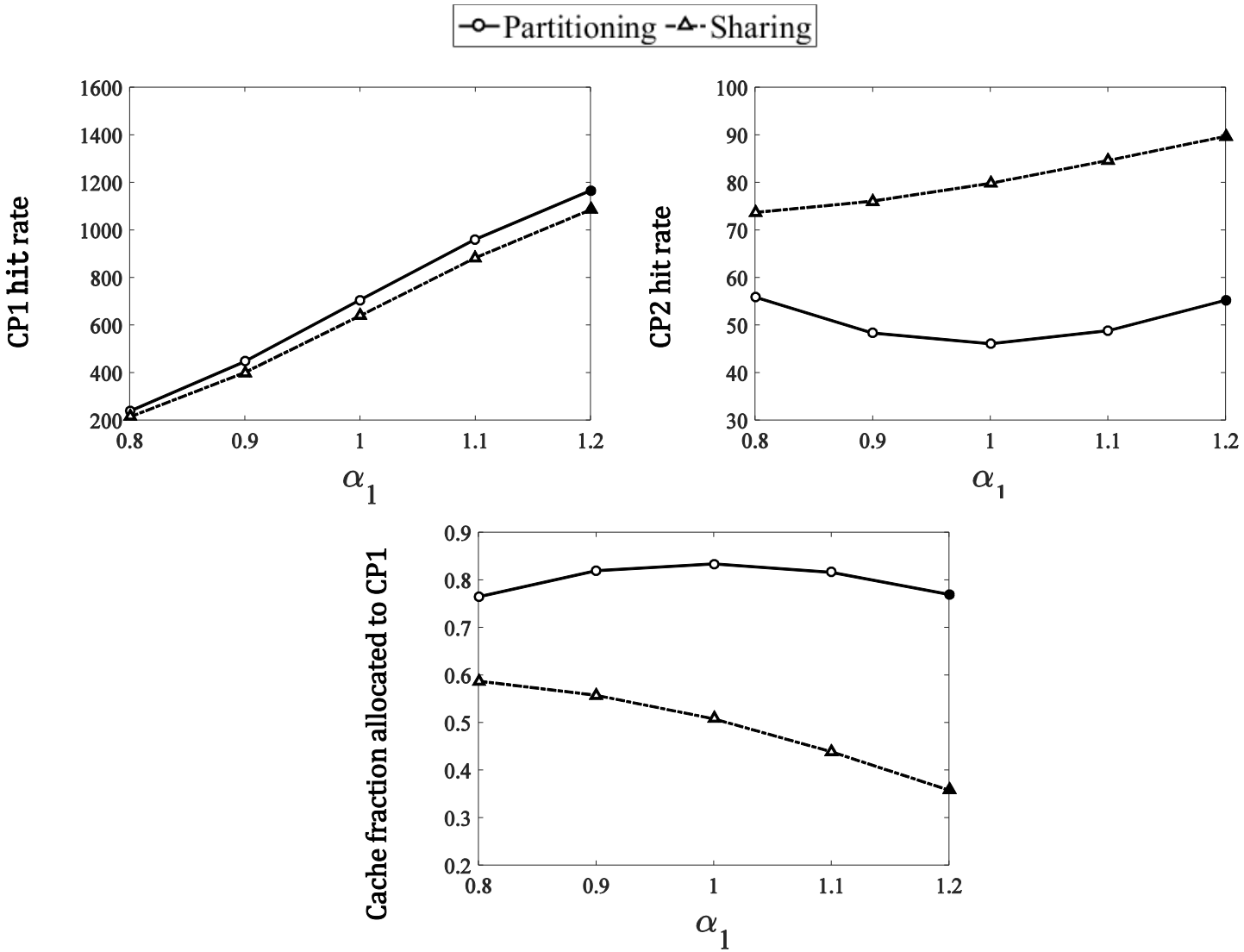
Parameter Impact of Request Rate



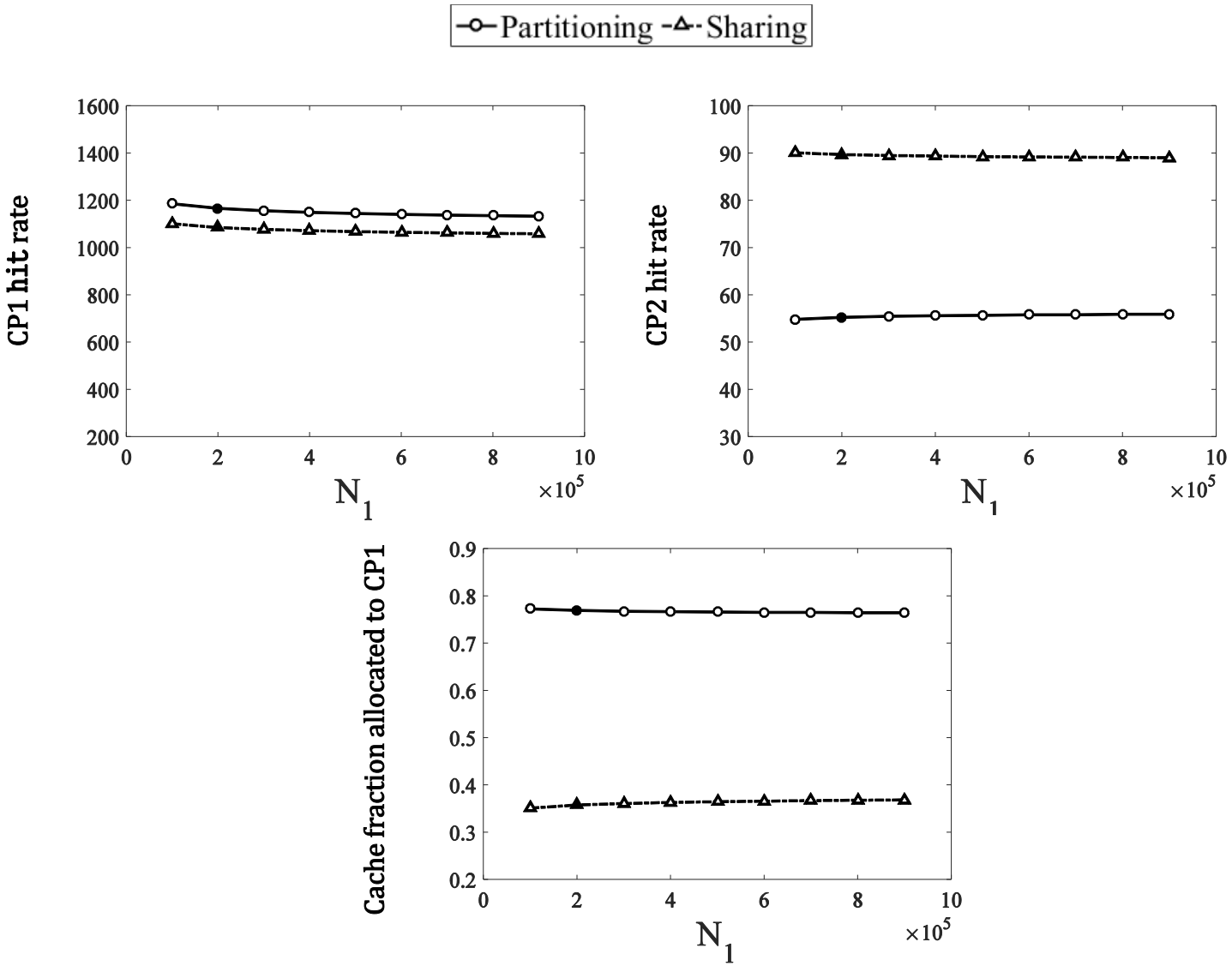
Parameter Impact of Weight



Parameter Impact of Skewness



Parameter Impact of File Population



Implications

- Use different utility functions to achieve different fairness: $U_k(h_k) = \frac{h_k^{1-\beta}}{1-\beta}$. $\beta \to 1$, *proportional* fairness; $\beta \to \infty$, *max-min* fairness;
- Online (decentralized) allocation: Kelly's framework.
- Develop adaptive control mechanism: in response to changing network traffic.
- Alternative formulation: delay optimization.
- Apply to policies other than LRU.

Thank you! Q & A ?