# Rethinking Web Cache Design for the AI Era

Yazhuo Zhang
ETH Zurich
Switzerland

Jinqing Cai
ETH Zurich
Switzerland

Avani Wildani
Cloudflare
United States

Ana Klimovic
ETH Zurich
Switzerland

## ABSTRACT

Web caches have long been effective at reducing latency and backend load by storing popular content close to users, exploiting the temporal and spatial locality of human-driven access patterns. However, the rise of AI-generated traffic is challenging this assumption. AI agents such as search crawlers and data scrapers issue large volumes of diverse, low-referrer requests with minimal reuse, which degrade cache effectiveness, interfere with human-relevant content, and increase pressure on backend systems. In this paper, we argue that caching infrastructure must evolve to address this shift. We analyze emerging AI traffic patterns and study their impact on caching performance using a CDN prototype based on Wikimedia's architecture. Our results show that even modest amounts of AI traffic lead to significant cache inefficiency. We envision a workload-aware caching paradigm that serves human and AI traffic through differentiated tiers and policies, preserving responsiveness for users while adapting to the diverse access patterns and requirements of AI workloads.

**ACM Reference Format:**
Yazhuo Zhang, Jinqing Cai, Avani Wildani, and Ana Klimovic. 2025. Rethinking Web Cache Design for the AI Era. In *ACM Symposium on Cloud Computing (SoCC '25), November 19–21, 2025, Online, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3772052.3772255

## 1 INTRODUCTION

Deployed across browsers, edge nodes, and content delivery networks (CDNs), web caches reduce latency and backend load by storing frequently accessed content closer to users [28, 40, 63]. Traditional cache design assumes that traffic exhibits strong temporal and spatial locality, an assumption that holds for human users, who often revisit popular or semantically related pages. This behavior results in skewed access distributions, where a small subset of content accounts for the majority of requests, making caching highly effective [6, 61].

This assumption, however, is increasingly invalid in the face of growing AI-generated web traffic. Automated bot traffic comprised an estimated 51% of web traffic in 2024, up from 43% in 2021, due to the rise of AI and Large Language Models (LLMs) [8, 30]. Cloudflare reports that 30% of its traffic originates from non-human agents, which includes HTTP request activities from user agents associated with AI assistants, AI data scrapers, and AI search crawlers [12]. Akamai similarly notes that AI scraper traffic is growing rapidly, already generating hundreds of millions of requests per day across their network [21]. Unlike human behavior, AI agents and scrapers exhibit aggressive behavior including high-volume, low-referrer requests, often in parallel and without regard for crawling norms [33, 49, 56]. Rather than focusing on popular pages, they frequently access rarely visited or loosely related content across a site, often in sequential, complete scans of the websites. For example, an AI assistant generating a response may fetch images, documentation, and knowledge articles across dozens of unrelated sources. While traditional web scrapers also exhibit sequential access patterns, their volume was historically limited, typically affecting only smaller, content-heavy websites [24, 46].

These shifts have real-world consequences. Traditional caching strategies are not designed to absorb the high diversity and low-reuse of traffic from AI scrapers and agents. As a result, backend systems face increased cold-path pressure, leading to higher compute and database load. Content providers such as Read the Docs have reported excessive backend service bandwidth consumption by AI scrapers — up to 73TB of HTML in a single month — resulting in significant financial costs and degraded performance [27]. Similarly, Wikimedia reports a 50% increase in backend service bandwidth usage from AI-driven scrapers. Although bots account for only 35% of Wikimedia's pageviews, they generate 65% of resource-consuming traffic (i.e., requests that bypass CDN cache layers and hit the core datacenters) [11, 56]. This reveals a fundamental limitation in today's caching infrastructure: it struggles to cope with increasingly mixed workloads from human and AI traffic.

In response, site operators have adopted two broad strategies to address AI traffic. Some treat AI bots as undesirable clients and employ defensive techniques such as bot filtering, rate limiting, `robots.txt` enforcement, and API access controls [9, 14, 48]. However, these approaches often fall short. Crawlers frequently rotate IPs and use legitimate user-agent strings, rendering IP and agent-based filters ineffective or overly aggressive, sometimes blocking real users [17, 27]. Others, recognizing the utility of AI agents in applications like search, summarization, and research, aim to serve them more effectively. For example, Cloudflare has introduced usage-based pricing for large-scale crawlers, signaling a shift toward usage accountability rather than outright blocking [49]. Yet this permissive approach highlights a growing need to optimize infrastructure for human and AI traffic, ensuring that human users continue to receive fast, reliable responses while accommodating the demands of automated agents.

In this paper, we argue that web caching systems should evolve to actively prioritize human traffic. Caches should not only act as passive stores optimized for popularity, but also become traffic-aware systems that recognize and adapt to the source and behavior of incoming requests. We analyze the behavior of both AI search crawlers and data scrapers, and evaluate cache performance under realistic workloads using a CDN prototype modeled after Wikimedia's infrastructure. We show that even modest levels of AI traffic


*SoCC '25, November 19–21, 2025, Online, USA*

ACM ISBN 979-8-4007-2276-9/2025/11
https://doi.org/10.1145/3772052.3772255

| System | AI Traffic Behavior | Impact | Mitigation |
|---|---|---|---|
| Wikimedia [11, 56] | bulk image scraping for model training | 50% surge in multimedia bandwidth usage | • block crawlers traffic |
| SourceHut [2, 10, 44] | LLM training crawler scraping code repos | service instability and slowdowns | • block crawlers traffic |
| Read the Docs [2, 27] | AI crawlers download large files hundreds of times daily | significant bandwidth increase | • IP-based rate limiting<br>• temporarily blocked all traffic from bots<br>• reconfigure CDN to better cache files |
| GNOME's GitLab [2] | burst of AI bot requests | service instability and slowdowns | • temporarily rate-limit on non-logged users<br>• block crawlers by Anubis [1] |
| KDE's GitLab [2] | aggressive scraping from AI bots mimicking browsers | service instability and slowdowns | • ban the specific Edge version used by bots |
| Fedora [2, 17, 39] | AI scrapers recursively crawl package mirrors | slowness for human users | • block a bunch of subnets<br>• geo-block traffic from known bot sources |
| Diaspora [43] | aggressive scraping without respecting robots.txt | downtime/slowness for human users | • rate limit<br>• block crawlers traffic |

**Table 1: Examples of infrastructure disruptions caused by AI traffic.**

can significantly degrade cache effectiveness, displacing human-relevant content and increasing backend load. We envision a new design paradigm where web caches first act as lightweight traffic filters, preserving locality by shielding human-driven requests from interference caused by high-volume, low-locality AI traffic. Additionally, since AI and human traffic exhibit fundamentally different access patterns and reuse behaviors, caches should separate their treatment, using distinct caching tiers or applying tailored admission and eviction policies to better serve each workload.

## 2 IS WEB CACHING STILL EFFECTIVE?

To understand how AI workloads challenge existing caching strategies, we begin by examining changes in web traffic composition and their impact through real-world evidence (§2.1). We then analyze the access patterns of two representative forms of AI traffic — AI search crawlers and AI data scrapers (§2.2). Finally, we present a prototype based on the Wikimedia CDN to to evaluate how AI-driven traffic affects cache performance (§2.3).

### 2.1 Shift in Web Traffic Patterns and its Impact

Web traffic composition is undergoing a major shift, with a growing share driven by AI systems rather than human users. Traditionally, web workloads follow a power-law distribution (also known as Zipfian), where a small fraction of objects receive the majority of requests [3–5, 15, 25, 26, 28, 45, 47, 59, 59, 62]. However, the rapid growth of AI-generated traffic is changing this landscape. AI-driven web traffic refers to automated requests generated by tools that support or train large-scale AI systems, including inference-serving agents (e.g., search assistants) and data scrapers used for model training [12, 18]. While bots have long existed on the web for tasks like search engine optimization (SEO) indexing, uptime monitoring, or price scraping, the recent surge of AI agents has introduced new traffic behaviors and scale [12, 18, 21, 49]. Recent data from Cloudflare indicates that over 30% of its global traffic now comes from non-human sources, including AI-powered assistants, search crawlers, and data scrapers [12]. Similarly, Dark Visitors, which monitors thousands of websites across the internet, reports that AI-related traffic rises from approximately 10% to 15% of total traffic just between April and July 2025 [18].

To quantify how aggressively AI platforms crawl the web, Cloudflare introduced the *crawl-to-refer* ratio, a metric that compares the number of HTML page requests made by a platform's automated agents to the number of actual user visits referred from that platform. A high crawl-to-refer ratio indicates that the platform crawls many pages, but few are ever shown to users, suggesting backend scraping activity rather than user-serving behavior. AI companies such as Anthropic and OpenAI exhibit extremely high ratios, $44,800\times$ and $1,300\times$, respectively, indicating that the vast majority of their crawled pages are never visited by users [49]. Similarly, Dark Visitors shows that only 2.19% of links mentioned in AI chat and search sessions are actually clicked by users [18]. This low referral click rate underscores a growing asymmetry: AI platforms aggressively retrieve content at scale, but only a small fraction is surfaced to or engaged with by end users.

For web caches, this results in a surge of one-time accesses to a wide range of unique pages over short time windows, leading to low hit rates and frequent evictions. Consequently, a large volume of traffic is offloaded to backend infrastructure — higher bandwidth usage, increased server compute demands, and costly database lookups. Table 1 summarizes real-world incidents where AI crawlers and scrapers have have caused service degradation and outages. The impact has been severe: Wikimedia experienced a 50% surge in multimedia bandwidth usage due to bulk image scraping, SourceHut, GNOME, and KDE reported service slowdowns and instability, Read the Docs noted significant bandwidth increases from AI bots repeatedly downloading large files, and Fedora and Diaspora suffered from heavy load and poor performance for human users [1, 2, 10, 11, 17, 27, 39, 43, 44, 56]. These disruptions reflect the strain that AI traffic, particularly from LLM training and

inference crawlers, can impose on infrastructure not designed for such workloads.

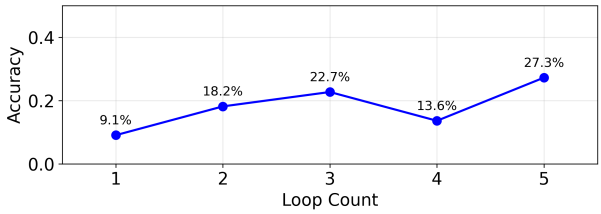
To mitigate the impact of AI crawlers, many platforms have resorted to defensive strategies such as blocking traffic, IP-based rate limiting, geo-blocking, or filtering by user-agent. However, these approaches often fall short. Crawlers frequently rotate IPs and use legitimate user-agent strings, rendering IP and agent-based filters ineffective or overly aggressive, sometimes blocking real users [17, 27]. Defensive tactics alone are not a long-term solution. As AI traffic continue to scale, it is no longer an anomaly; it's becoming a dominant mode of access. In response, Cloudflare introduced a pay-per-crawl model for AI bots [13], arguing that the infrastructure costs of serving automated crawlers should be compensated. At the same time, this shift calls for a deeper rethinking of infrastructure design — not just how to restrict AI traffic, but how to support it more effectively.

## 2.2 AI Traffic Patterns

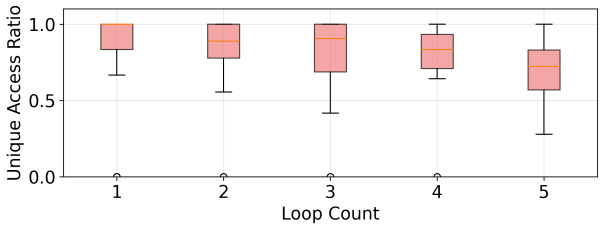
In this work, we focus on two representative categories of AI-driven traffic: AI search crawlers and AI data scrapers, as they have emerged as the most active bot types in recent analyses [33]. AI search crawlers fetch content to support live AI services, such as answering questions or summarizing pages, while AI data scrapers focus on harvesting data to build large training corpora for models like LLMs. Both generate large volumes of requests and access web content in disruptive ways, as we characterize below.

**AI Search Crawler.** To characterize the access pattern of AI search crawlers, we run an experiment using the LangChain Local Deep Researcher [34] to simulate an AI agent answering real-world information-seeking questions in the GAIA benchmark [23], which consists of over 450 non-trivial questions with unambiguous answers. To focus specifically on web search behavior, we extract 22 questions that require only access to search engines, excluding those that depend on external tools such as calculators, PDF readers, or Python compilers. The Local Deep Researcher agent conducts web-based research by iteratively querying a search engine, retrieving content, and refining its response with each loop. We use the DuckDuckGo search engine to execute each query, with responses generated using the Qwen3-8B model. While this experiment simulates a single agent, the behavior it exhibits is representative of emerging AI search patterns. Our goal is not to model aggregate traffic volumes, but to highlight structural trends, such as low content reuse and broad page coverage, that can affect caching and backend systems.

Figure 1 shows the agent's accuracy and unique access ratio as a function of the number of search-and-retrieval loops. In each loop, the agent issues a query and retrieves three results. Increasing the number of loops expands the agent's access to potentially relevant content, but also increases total traffic volume. We vary the loop count to understand how iterative retrieval influences both task quality (measured by accuracy) and traffic behavior (measured by unique access ratio). To assess how efficiently the agent utilizes this traffic, we define the unique access ratio as the fraction of page requests within a single GAIA task that target previously unseen pages in that task. Across all loop counts, the unique access ratio




**(a) Accuracy: the number of correct answers over 22 tasks.**




**(b) Unique access ratio: the distribution of unique URL requests across tasks. High ratios (70–100%) indicate low content reuse and broad coverage.**

**Figure 1: Accuracy and unique access ratio of Local Deep Researcher on 22 GAIA benchmark tasks, evaluated over increasing search loop counts using DuckDuckGo. Each loop issues a query and retrieves three results.**

remains consistently high, typically between 70% and 100%, indicating that the agent fetches mostly new content rather than revisiting previously seen pages. Meanwhile, accuracy improves with more loops, increasing from 9.1% with one loop to 27.3% with five loops. The result aligns with the results from the GAIA leaderboard on HuggingFace, where the Qwen3-32B family achieves an average score of 21.59% [29]. These results suggest that AI search agents rely on accessing a broad set of unique pages to achieve better task performance, and that iterative retrieval contributes both to improved accuracy and to high-volume, low-reuse traffic patterns.

Notably, our analysis of the fetched URLs reveals that the most frequent accessed domain was `en.wikipedia.org`. As one of the largest content providers, Wikipedia has been significantly impacted by AI traffic, as discussed earlier. In §2.3, we examine how AI traffic degrades the effectiveness of Wikimedia's edge caching infrastructure.

**AI Data Scraper.** Unlike AI search crawlers that operate interactively and incrementally, data scrapers often traverse the web in bulk, issuing high-throughput requests across vast content domains. We highlight three characteristics of AI scraper traffic: high unique URL ratio, content diversity, and crawling inefficiency.

First, AI scrapers exhibit high unique access ratios. Public crawl statistics from Common Crawl, which performs large-scale legitimate web crawls on a monthly basis, show that the vast majority of fetched pages are unique [16]. Our experiments using Crawl4AI [50], a web crawling framework designed specifically for LLMs, AI agents, and data pipelines, confirm that across various search strategies (e.g., breadth-first, depth-first, best-first), the unique URL ratio consistently approaches one. Although scrapers

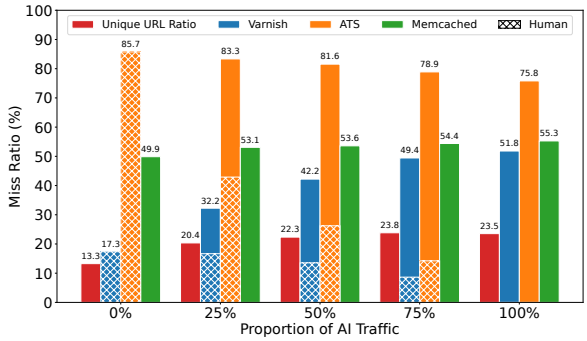



**Figure 2: Wikimedia cache miss ratios across Varnish, ATS, and Memcached under varying proportions of AI and human traffic, along with unique URL ratios. AI and human workloads share about 20% overlapping content.**

may occasionally revisit known content, they tend to do so infrequently. From the perspective of a web server, this behavior still appears highly aggressive.

Second, AI scrapers show strong diversity in content selection. Studies [21, 52] reveal that different AI crawlers target distinct content types, some specialize in technical documentation (.md, .pdf, .txt), while others focus on source code, media, or blog posts. This variation creates uneven load distribution across the web, with some domains (e.g., Wikipedia, GitHub, academic repositories) becoming hotspots of scraper activity. The semantic skew of AI traffic implies that web infrastructure must handle not only high volumes but also concentrated pressure on specific content verticals.

Third, AI scrapers suffer from crawling inefficiency. Unlike mature search engine crawlers that carefully optimize URL selection and revisit strategies, many AI data scrapers still generate a high volume of low-value or failed requests. Recent analysis [52] report that a substantial fraction of fetches from popular AI crawlers result in 404 errors or redirects, often due to poor URL handling. The rate of these ineffective requests varies depending on how well the crawler is optimized to target live, meaningful content. Such inefficiencies not only reduce the quality of the collected data but also waste significant bandwidth and impose avoidable load on origin servers.

All together, both AI search crawlers and data scrapers generate traffic that exhibits a *scan pattern*: a request stream that touches a large number of distinct objects with little-to-no reuse in a short period. In the context of storage and database systems, workloads with scan pattern are known to degrade caching efficiency, as they overwhelm limited cache capacity and prevent stable reuse [7, 20, 31, 32, 38, 42, 60].

## 2.3 Case Study: Wikimedia CDN

To measure the performance impact of evolving access patterns in web caches, we prototype a CDN architecture based on Wikimedia's production CDN. As discussed in §2.2, scan access pattern is a good fit for representing AI traffic. We simulate human and AI traffic using Zipfian and scan benchmarks, respectively.

**CDN Architecture.** Wikimedia's production CDN features a globally distributed, two-tier caching hierarchy: a frontend cache layer using Varnish, and a backend disk-based cache layer using Apache Traffic Server (ATS). These cache nodes are deployed across multiple data centers worldwide, organized into core and edge sites. Core data centers house origin servers and persistent storage, while edge cache sites serve as entry points for user traffic, reducing latency and offloading backend systems. The caching infrastructure is backed by MediaWiki application servers, Memcached for metadata and template fragments, and MariaDB for persistent content storage [41, 55, 57].

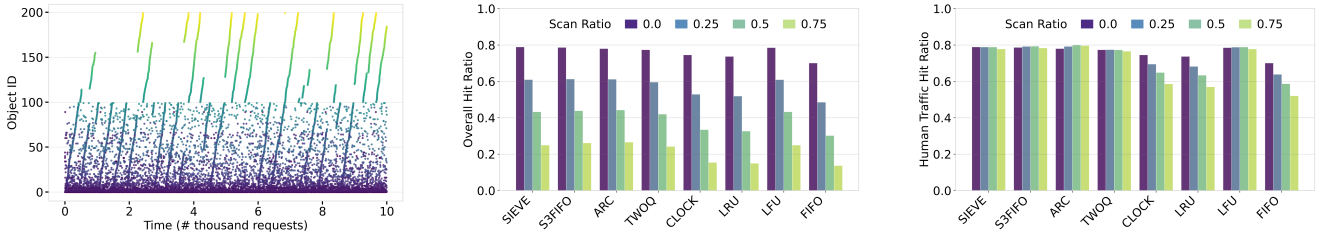
**Prototype.** Our prototype replicates Wikimedia's multi-tier caching architecture, with Varnish and ATS serving as HTTP accelerators and frontend caches, and Memcached acting as the backend metadata store. Request handling follows Wikimedia's cache pipeline. All incoming HTTP GET requests are first processed by Varnish, which caches full HTML responses for users. On a miss, the request is forwarded to ATS. If the requested content is still not found, ATS forwards the request to the MediaWiki application server. MediaWiki then checks Memcached to retrieve any cached intermediate artifacts, such as pre-parsed templates, Lua module output, page metadata, and link tables, that can accelerate rendering [37, 53]. If the necessary data is not found in Memcached, MediaWiki queries MariaDB to fetch the raw wikitext, renders it into HTML, and returns the response. The final rendered HTML is then cached in Varnish and delivered to the client.

**Experimental setup.** We deploy the system on a CloudLab [19] r6525 node with dual AMD EPYC 7543 CPUs (128 hardware threads). The stack is containerized using Docker, with CPU allocations manually tuned for each service. The Varnish cache is configured with 256 MB and ATS with 512 MB of memory. Wikipedia content is imported using MediaWiki's importDump.php tool and a selected XML stream from the 2025-05-20 enwiki dump [22].

Traffic is generated using Locust [35], with two distinct client models: (1) Human users follow a Zipfian access distribution ($\alpha$ = 1.0), focusing requests on a small subset of popular pages; and (2) AI users emulate scan-style workloads using a depth-first traversal of internal links, resulting in broad, low-locality access patterns

**Results.** Figure 2 shows cache miss ratios across Varnish, ATS, and Memcached as the proportion of AI traffic increases, together with the fraction of unique URLs accessed. When traffic is purely human-like, the Varnish miss ratio is low (17.3%), indicating efficient caching. However, even modest amounts of AI traffic cause a pronounced degradation: with just 25% AI traffic, the miss rate nearly doubles to 32.2%. As the AI share continues to grow, cache locality deteriorates rapidly, and the Varnish miss ratio reaches 51.8% under fully scan-dominated workloads. This degradation corresponds closely with the rise in unique URLs requested. From the miss breakdown by traffic type, we see that AI requests contribute disproportionately to total misses. Even with a balanced mix (50% AI / 50% human), more than two-thirds of Varnish misses originate from AI requests, showing that AI traffic drives cache churn and evicts content valuable to human users.

Interestingly, ATS's miss ratio decreases as the AI fraction increases because it absorbs the rising number of Varnish misses. Under AI workloads, many short-term revisits and overlapping requests still miss in Varnish but hit in ATS, whose larger capacity




**(a) Mixed traffic combining human and AI access patterns. Each dot represents an unique requested object over time, and colors distinguish object IDs. Diagonal lines indicate sequential scan access.**

**(b) Overall cache hit ratio decreases sharply with increasing scan ratio, indicating higher backend load.**

**(c) Human traffic hit ratio across different eviction algorithms under varying AI scan ratios. Smarter policies maintain high hit ratio for human traffic even under interference.**

**Figure 3: Impact of mixed workloads on caching behavior.**

retains full HTML pages longer. By contrast, Memcached's miss ratio slightly increases as AI traffic rises. Broader, low-locality AI access triggers a larger variety of page templates and metadata lookups in MediaWiki, expanding Memcached's working set and evicting previously hot entries. The increased Varnish and ATS misses also push more requests toward MediaWiki, further stressing Memcached.

Overall, assuming AI traffic follows a scan pattern while human traffic presents a Zipfian distribution, these results show that AI traffic not only increases misses at the edge but also alters the dynamics of downstream caches, underscoring the need for cache hierarchies that explicitly differentiate traffic types and isolate AI-driven workloads to sustain performance for human users.

## 3 RETHINKING WEB CACHE DESIGN

In this section, we outline our vision for cache systems that adapt to the growing mix of human and AI workloads. We argue that caches should evolve along two complementary directions.

In §3.1, we explore how caches can act as traffic filters, prioritizing reuse-heavy human requests and limiting the disruption caused by low-locality AI traffic. We show that advanced eviction algorithms can naturally serve this role, even without explicit traffic classification.

In §3.2, we shift from algorithmic mechanisms to architectural cache design that explicitly separates human and AI traffic across tiers and policies. This separation aims to better align caching strategies with the distinct characteristics of each workload, improving efficiency, reducing backend load, and preserving service quality as AI traffic continues to grow.

### 3.1 From Storage Layer to Traffic Filter

Traditional caches treat all traffic equally, but AI traffic can overwhelm cache capacity and displace valuable human-serving content. We hypothesize that advanced eviction algorithms (e.g., SIEVE, S3FIFO, ARC) can act as implicit traffic filters, deprioritizing scan-like AI requests while preserving high hit rates for human traffic. We test this hypothesis by running the following experiment.

We simulate mixed workloads that combine Zipfian (human-like) and scan (AI-like) access patterns. Figure 3a visualizes a sample request trace where 25% of the traffic follows a sequential scan pattern (AI), while the remaining 75% follows a skewed Zipfian distribution (human). Each dot represents an object request over time, with colors distinguish object IDs. The diagonal lines correspond to AI requests sequentially scanning new content, while the dense region in the bottom reflects human traffic following a Zipfian distribution, where a small number of popular objects account for most accesses. Besides, the scan pattern intersecting the dense region in the lower part of the figure means that AI and human traffic can target the same popular content.

We quantify performance using two metrics: the overall cache hit ratio, which measures the fraction of all requests (human and AI) served from cache, and the human traffic hit ratio, defined as the hit rate observed specifically on requests originating from human traffic. While overall hit ratio reflects aggregate cache performance, human traffic hit ratio captures how well the cache continues to serve human users in the presence of interfering AI traffic.

We evaluate state-of-the-art cache eviction algorithms under mixed workloads using a synthetic trace of 10 million requests over 10k unique objects. Human traffic follows a Zipfian distribution ($\alpha$ = 1.0), AI traffic performs sequential scans, with 10% overlap between them. The cache size is configured as 10% of the working set. Figure 3b shows that the overall cache hit ratio drops significantly as the scan ratio increases from 0 to 75% across all evaluated eviction algorithms. Under pure Zipfian traffic, most algorithms achieve 70–80% hit rates, with SIEVE, S3FIFO, ARC, and LFU performing the best. However, as the fraction of AI traffic increases, hit ratios decline steeply, falling below 30% for all policies at a 75% scan ratio. While overall hit ratio reflects aggregate cache performance, it can obscure how well the cache continues to serve latency-sensitive human traffic in the presence of interfering AI access. As shown in Figure 3c, several state-of-the-art algorithms are able to preserve high hit ratios for human traffic. These policies maintain over 75% hit ratio for human requests even when scan ratio reaches 50%, whereas simpler strategies like LRU, LFU, and FIFO degrade more quickly. These advanced algorithms are effective at preserving human traffic due to their ability to quickly demote unpopular objects [58]. For example, S3FIFO and TwoQ use a small FIFO queue to quickly identify and evict one hit wonders [36, 54]; SIEVE uses a moving hand to quickly remove unpopular object – the scan accesses in this case. While previous studies have evaluated these

algorithms for overall cache hit ratios, their effectiveness in mixed workloads has not been thoroughly explored.

Our results show that smarter eviction algorithms can serve as lightweight traffic filters: even under substantial AI interference, policies like SIEVE, S3FIFO, and ARC preserve high hit ratios for human traffic. While frontend mechanisms, such as rate limiters or bot detection, can help identify aggressive AI traffic, advanced cache eviction policies offer an additional layer of protection. However, simply bypassing AI traffic to the backend is not ideal as it increases pressure on application servers, storage systems, and databases. In the next section, we explore how to better accommodate AI requests through alternative caching strategies and system design.

## 3.2 Composable Cache Architectures

Complementary to making existing caches more resilient, we propose a composable caching hierarchy that coordinates differentiated tiers for human and AI workloads while enabling selective content sharing across layers. Instead of strict separation, this design allows overlapping cache tiers with distinct roles and policies but shared intermediate representations.

To balance latency, capacity, and cost, human traffic should continue to be served primarily from edge caches optimized for low latency and high hit ratios. These caches prioritize responsiveness and can adopt scan-resistant eviction to preserve locality. Interactive AI crawlers (e.g., retrieval-augmented generation) require moderate latency and can be routed to mid-tier caches such as ATS, which balance capacity and responsiveness. Bulk AI scrapers (e.g., training corpus builders) can tolerate higher latency and be handled by deeper, cost-efficient caches near the origin, possibly with delayed admission or rate-limiting.

Across these tiers, composable caching layers store overlapping content shared by human and AI workloads. Overlap can be identified by tracking objects frequently requested by both traffic types, promoting them to the shared tier while routing exclusive content to separate caches. This shared layer retains commonly accessed objects, reducing duplication and backend load.

Each tier applies policies suited to its traffic class. Human-facing caches prioritize freshness and reuse, using short TTLs and aggressive invalidation to reflect the latest content. AI-facing caches emphasize coverage and efficiency, using longer TTLs, lazy invalidation, and coarse-grained batching to tolerate staleness. Logical partitioning (e.g., per-tenant slices) further isolates diverse AI agents or task types, while selective sharing of popular content maximizes reuse across workloads.

# 4 DISCUSSION AND OPEN CHALLENGES

While composable cache hierarchies provide a promising foundation, several open questions remain before such systems can be deployed at scale.

**How can we reliably identify AI traffic?** Accurate traffic identification underpins any differentiated caching strategy. Existing methods rely on user-agent strings, IP-based heuristics, or behavioral signatures, but these are unreliable as modern AI agents can disguise themselves behind legitimate browsers or proxy endpoints.

A key question is whether caches can infer AI intent from access dynamics such as reuse ratio, traversal depth, or burstiness. In the longer term, we envision cooperative identification through standards like the Model Context Protocol (MCP), where AI systems explicitly tag their requests with purpose, priority, and rate preferences, allowing infrastructure to enforce cache-aware policies without adversarial detection.

**How should caches handle different AI traffic subclasses?** AI traffic is highly diverse, including bulk crawlers that collect training data, iterative search agents that fetch related documents in short bursts, and interactive assistants that retrieve context on demand. Each subclass exerts distinct pressure on the cache hierarchy. Bulk scrapers generate large volumes of unique objects with low reuse and can tolerate higher latency; such requests could be redirected to deep, rate-limited cache tiers. Search agents display moderate locality and can benefit from mid-tier caches with predictive prefetching and longer TTLs. Interactive assistants demand low latency and freshness and should share optimized edge caches with human traffic. A key research direction is developing workload-adaptive cache controllers that detect these modes automatically and adjust TTLs, eviction priorities, or throttling thresholds in real time.

**How can caches express their objectives declaratively?** Today's caches optimize hidden metrics such as hit ratio or latency without exposing their underlying trade-offs. Future systems should instead support declarative cache objectives, allowing operators or applications to specify explicit goals, such as minimize human response latency, cap AI traffic bandwidth, or maximize reuse under resource limits. These objectives can drive adaptive cache controllers that continuously tune admission, eviction, and refresh policies based on observed workload conditions. By aligning control decisions with stated intent, caches can evolve from heuristic-driven mechanisms into goal-oriented systems that balance freshness, reuse, and fairness across human and AI workloads.

**How to design fair, predictable pricing that sustains openness?** AI activity risks undermining the web's economic foundations [51]. Serving AI traffic consumes significant storage, bandwidth, and CPU resources while offering limited direct benefit to content providers. How should cache usage be priced to reward cache-efficient agents that reuse content and respect rate limits, while discouraging redundant large-scale scraping? CDNs could introduce usage-tiered pricing or priority access levels. The broader challenge lies in balancing economic sustainability and openness, ensuring that caching infrastructure remains publicly accessible while distributing the cost of AI-driven load fairly among actors in the web ecosystem.

# 5 CONCLUSION

AI traffic is fundamentally reshaping web caching. As low-locality traffic from AI crawlers and scrapers grows, traditional caches struggle to preserve performance for human users. We argue for workload-aware cache designs that differentiate and coordinate resources across traffic types. Using scan-pattern benchmarks and a Wikimedia CDN case study, we show how AI workloads disrupt locality and motivate tiered placement and differentiated policies. Future caches must tackle challenges in traffic identification, workload classification, and fair pricing to sustainably support both human and AI access at scale.

# REFERENCES


[1] Anubis. Anubis: Web ai firewall utility. https://anubis.techaro.lol/. Accessed: 2025-07-11.

[2] Daniel Atherton. Incident 1001: Llm scrapers allegedly target multiple open source projects disrupting the foss ecosystem. https://incidentdatabase.ai/cite/1001/. Accessed: 2025-07-11.

[3] Benjamin Berg, Daniel S. Berger, Sara McAllister, Isaac Grosof, Sathya Gunasekar, Jimmy Lu, Michael Uhlar, Jim Carrig, Nathan Beckmann, Mor Harchol-Balter, and Gregory R. Ganger. The CacheLib caching engine: Design and experiences at scale. In *14th USENIX symposium on operating systems design and implementation (OSDI 20)*, OSDI'20, pages 753–768. USENIX Association, November 2020.

[4] L. Breslau, Pei Cao, Li Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: evidence and implications. In *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, pages 126–134 vol.1, New York, NY, USA, 1999. IEEE.

[5] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. On the implications of Zipf's law for web caching. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 1998.

[6] Lee Breslau, Pei Cao, Li Fan, Graham Phillips, and Scott Shenker. Web caching and zipf-like distributions: Evidence and implications. In *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, volume 1, pages 126–134. IEEE, 1999.

[7] Marc Brooker. Why aren't we sieve-ing? https://brooker.co.za/blog/2023/12/15/sieve.html, 2023. Accessed: 2025-07-13.

[8] Business Wire. Artificial intelligence fuels rise of hard-to-detect bots that now make up more than half of global internet traffic, according to the 2025 imperva bad bot report. https://www.businesswire.com/news/home/20250415432215/en/Artificial-Intelligence-Fuels-Rise-of-Hard-to-Detect-Bots-That-Now-Make-up-More-Than-Half-of-Global-Internet-Traffic-According-to-the-2025-Imperva-Bad-Bot-Report, 2025. Accessed: 2025-05-21.

[9] Fabrice Canel and Krishna Madhavan. Robots exclusion protocol extension to communicate ai preferences vocabulary. https://datatracker.ietf.org/doc/draft-canel-robots-ai-control/, 2025. Accessed: 2025-07-06.

[10] Thomas Claburn. Ai crawlers haven't learned to play nice with websites. https://www.theregister.com/2025/03/18/ai_crawlers_sourcehut/. Accessed: 2025-07-11.

[11] Thomas Claburn. Wikipedia's overlords bemoan ai bot bandwidth burden. https://www.theregister.com/2025/04/03/wikimedia_foundation_bemoans_bot_bandwidth/#:~:text=That%27s%20due%20to%20the%20Wikimedia,which%20consumes%20more%20computing%20resources, April 2025. Accessed: 2025-06-17.

[12] Cloudflare. Cloudflare radar: Global internet traffic. https://radar.cloudflare.com/traffic, April 2025. Accessed: 2025-05-21.

[13] Cloudflare. Introducing pay-per-crawl: Making ai bots accountable for their resource use. https://blog.cloudflare.com/introducing-pay-per-crawl/, 2025. Accessed: 2025-07-07.

[14] Cloudflare. Navigating the ai labyrinth: Understanding and managing ai bot traffic. https://blog.cloudflare.com/ai-labyrinth/, 2025. Accessed: 2025-07-07.

[15] Brian F Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 143–154, 2010.

[16] Common Crawl. Common crawl statistics dashboard. https://commoncrawl.github.io/cc-crawl-statistics/plots/crawlsize, 2024. Accessed: 2025-07-11.

[17] Cryptodamus. Ai web scrapers attacking open source — here's how to fight back. https://cryptodamus.io/en/articles/news/ai-web-scrapers-attacking-open-source-here-s-how-to-fight-back. Accessed: 2025-07-11.

[18] Dark Visitors. Ai web traffic insights. https://darkvisitors.com/insights, 2025. Accessed: 2025-07-07.

[19] Dmitry Duplyakin, Robert Ricci, Aleksander Maricq, Gary Wong, Jonathon Duerig, Eric Eide, Leigh Stoller, Mike Hibler, David Johnson, Kirk Webb, et al. The design and operation of {CloudLab}. In *2019 USENIX annual technical conference (USENIX ATC 19)*, pages 1–14, 2019.

[20] Gil Einziger, Roy Friedman, and Ben Manes. Tinylfu: A highly efficient cache admission policy. *ACM Transactions on Storage (ToS)*, 13(4):1–31, 2017.

[21] Tom Emmons and Robert Lester. The rise of the llm ai scrapers: What it means for bot management. https://www.akamai.com/blog/security/rise-llm-ai-scrapers-bot-management, 2025. Accessed: 2025-07-06.

[22] Wikimedia Foundation. English wikipedia dump: 2025-05-20. https://dumps.wikimedia.org/enwiki/20250520/, 2025. Accessed: 2025-07-11.

[23] GAIA Benchmark Contributors. Gaia benchmark leaderboard. https://huggingface.co/spaces/gaia-benchmark/leaderboard, 2024. Accessed: 2025-07-06.

[24] C Lee Giles, Yang Sun, and Isaac G Councill. Measuring the web crawler ethics. In *Proceedings of the 19th international conference on World wide web*, pages 1101–1102, 2010.

[25] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti. Youtube traffic characterization: a view from the edge. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 15–28, 2007.

[26] Syed Hasan, Sergey Gorinsky, Constantine Dovrolis, and Ramesh K Sitaraman. Trade-offs in optimizing the cache deployments of cdns. In *IEEE INFOCOM 2014-IEEE conference on computer communications*, pages 460–468. IEEE, 2014.

[27] Eric Holscher. Ai crawlers need to be more respectful. https://about.readthedocs.com/blog/2024/07/ai-crawlers-abuse/, 2025. Accessed: 2025-07-03.

[28] Qi Huang, Ken Birman, Robbert van Renesse, Wyatt Lloyd, Sanjeev Kumar, and Harry C. Li. An analysis of Facebook photo caching. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, pages 167–181, New York, NY, USA, November 2013. Association for Computing Machinery.

[29] HuggingFace. Gaia benchmark leaderboard. https://huggingface.co/spaces/gaia-benchmark/leaderboard, 2024. Accessed: 2025-07-11.

[30] Imperva. 2025 bad bot report: The rise of ai-powered automation. Technical report, 2025. Accessed: 2025-07-12.

[31] Song Jiang and Xiaodong Zhang. Lirs: An efficient low inter-reference recency set replacement policy to improve buffer cache performance. *ACM SIGMETRICS Performance Evaluation Review*, 30(1):31–42, 2002.

[32] Theodore Johnson and Dennis Shasha. 2q: A low overhead high performance buffer management replacement algorithm. In *Proceedings of the 20th VLDB Conference*, pages 439–450, 1994.

[33] Taein Kim, Karstan Bock, Claire Luo, Amanda Liswood, and Emily Wenger. Scrapers selectively respect robots. txt directives: evidence from a large-scale empirical study. *arXiv preprint arXiv:2505.21733*, 2025.

[34] LangChain AI. Local deep researcher. https://github.com/langchain-ai/local-deep-researcher, 2024. Accessed: 2025-07-06.

[35] Locust. Locust: Scalable user load testing tool. https://locust.io/, 2024. Accessed: 2025-07-06.

[36] Bruce M Maggs and Ramesh K Sitaraman. Algorithmic nuggets in content delivery. *ACM SIGCOMM Computer Communication Review*, 45(3):52–66, 2015.

[37] MediaWiki. Manual:object cache. https://www.mediawiki.org/wiki/Object_cache, 2025. Accessed: 2025-07-11.

[38] Nimrod Megiddo and Dharmendra S Modha. ARC: A self-tuning, low overhead replacement cache. In *2nd USENIX conference on file and storage technologies (FAST 03)*, FAST'03, 2003.

[39] nirik. Mid march infra bits 2025. https://www.scrye.com/blogs/nirik/posts/2025/03/15/mid-march-infra-bits-2025/. Accessed: 2025-07-11.

[40] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The Akamai network: a platform for high-performance internet applications. *ACM SIGOPS Operating Systems Review*, 44(3):2–19, August 2010.

[41] Emanuele Rocca. Wikimedia's cdn up to 2018: Varnish and ipsec. https://techblog.wikimedia.org/2020/10/14/wikimedias-cdn/, 2020. Accessed: 2025-07-13.

[42] Liana V Rodriguez, Farzana Yusuf, Steven Lyons, Eysler Paz, Raju Rangaswami, Jason Liu, Ming Zhao, and Giri Narasimhan. Learning cache replacement with {CACHEUS}. In *19th USENIX Conference on File and Storage Technologies (FAST 21)*, pages 341–354, 2021.

[43] Dennis Schubert. Post noting 70% of server traffic from llm training bots. https://diaspo.it/posts/2594. Accessed: 2025-07-11.

[44] SourceHut Status. Incident report: git.sr.ht degraded by llm crawlers. https://status.sr.ht/issues/2025-03-17-git.sr.ht-llms/. Accessed: 2025-07-11.

[45] Kunwadee Sripanidkulchai, Bruce Maggs, and Hui Zhang. An analysis of live streaming workloads on the internet. In *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 41–54, 2004.

[46] Yang Sun, Isaac G Councill, and C Lee Giles. The ethicality of web crawlers. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 668–675. IEEE, 2010.

[47] Aditya Sundarrajan, Mingdong Feng, Mangesh Kasbekar, and Ramesh K Sitaraman. Footprint descriptors: Theory and practice of cache provisioning in a global cdn. In *Proceedings of the 13th International Conference on emerging Networking EXperiments and Technologies*, pages 55–67, 2017.

[48] Cloudflare Radar Team. Declaring your aindependence: Block ai bots, scrapers and crawlers with a single click. https://blog.cloudflare.com/declaring-your-aindependence-block-ai-bots-scrapers-and-crawlers-with-a-single-click/, 2024. Accessed: 2025-07-07.

[49] Cloudflare Radar Team. The crawl-to-refer ratio on radar ai insights. https://blog.cloudflare.com/ai-search-crawl-refer-ratio-on-radar/, 2025. Accessed: 2025-07-06.

[50] unclecode. Crawl4ai: A configurable web crawler for ai workload simulation. https://github.com/unclecode/crawl4ai, 2024. Accessed: 2025-07-11.

[51] Moshe Y Vardi. Will ai destroy the world wide web? *Communications of the ACM*, 68(8):5–5, 2025.

[52] Vercel. The rise of the ai crawler. https://vercel.com/blog/the-rise-of-the-ai-crawler. Accessed: 2025-07-11.

[53] Wikemedia. Lua scripting in mediawiki. https://www.mediawiki.org/wiki/Lua/Overview, n.d. Accessed: 2025-07-14.

[54] Wikimedia. Better handling for one-hit-wonder objects. https://phabricator.wikimedia.org/T144187, 2016. Accessed: 2025-07-14.

[55] Wikimedia Foundation. Data centers. https://wikitech.wikimedia.org/wiki/Data_centers, 2025. Accessed: 2025-07-13.

[56] Wikimedia Foundation. How crawlers impact the operations of the wikimedia projects. https://diff.wikimedia.org/2025/04/01/how-crawlers-impact-the-operations-of-the-wikimedia-projects/, 2025. Accessed: 2025-07-07.

[57] Wikimedia Foundation. Wikimedia cdn. https://wikitech.wikimedia.org/wiki/CDN, 2025. Accessed: 2025-07-13.

[58] Juncheng Yang, Ziyue Qiu, Yazhuo Zhang, Yao Yue, and KV Rashmi. Fifo can be better than lru: the power of lazy promotion and quick demotion. In *Proceedings of the 19th Workshop on Hot Topics in Operating Systems*, pages 70–79, 2023.

[59] Juncheng Yang, Yao Yue, and K. V. Rashmi. A large scale analysis of hundreds of in-memory cache clusters at Twitter. In *14th USENIX symposium on operating systems design and implementation (OSDI 20)*, OSDI'20, pages 191–208. USENIX Association, November 2020.

[60] Juncheng Yang, Yazhuo Zhang, Ziyue Qiu, Yao Yue, and K.V. Rashmi. Fifo queues are all you need for cache eviction. In *Symposium on Operating Systems Principles (SOSP'23)*, 2023.

[61] Yue Yang and Jianwen Zhu. Write skew and zipf distribution: Evidence and implications. *ACM transactions on Storage (TOS)*, 12(4):1–19, 2016.

[62] Yazhuo Zhang, Juncheng Yang, Yao Yue, Ymir Vigfusson, and KV Rashmi. {SIEVE} is simpler than {LRU}: an efficient {Turn-Key} eviction algorithm for web caches. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 1229–1246, 2024.

[63] Ke Zhou, Si Sun, Hua Wang, Ping Huang, Xubin He, Rui Lan, Wenyan Li, Wenjie Liu, and Tianming Yang. Demystifying Cache Policies for Photo Stores at Scale: A Tencent Case Study. In *Proceedings of the 2018 International Conference on Supercomputing*, pages 284–294, Beijing China, June 2018. ACM.