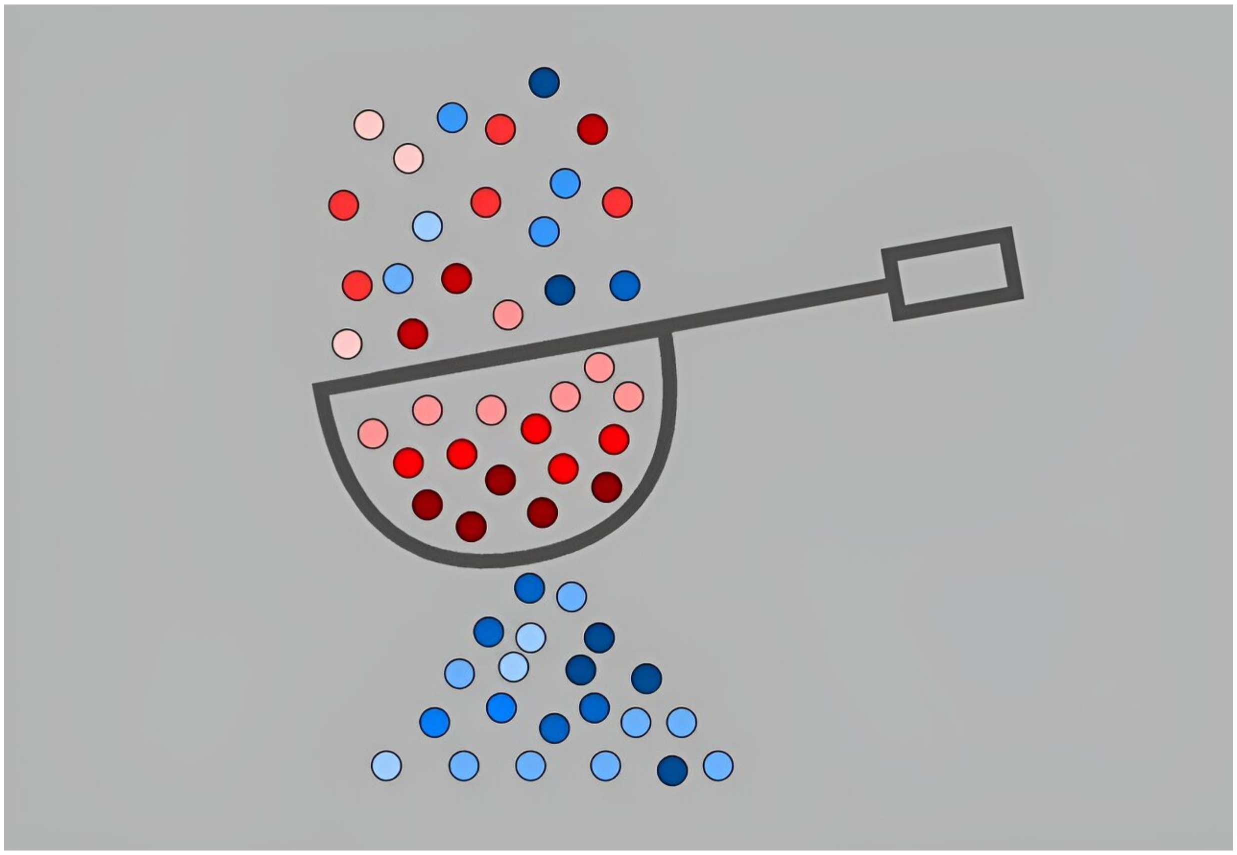
Computer scientists invent simple method to speed cache sifting

January 24 2024, by Carol Clark



A logo for SIEVE, created by Yazhuo Zhang, portrays hotter objects in shades of red and colder ones in shades of blue. Zhang also designed a web site for SIEVE, including a motion graphic demonstrating how it works. Credit: Yazhuo Zhang

Computer scientists have invented a highly effective—yet incredibly simple—algorithm to decide which items to toss from a web cache to make room for new ones. Known as SIEVE, the new open-source algorithm holds the potential to transform the management of web traffic on a large scale.

SIEVE is a joint project of computer scientists at Emory University, Carnegie Mellon University and the Pelikan Foundation. The team's paper on SIEVE will be presented at the 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI) in Santa Clara, California in April.

A preprint of the paper is already making waves.

"SIEVE is bigger and greater than just us," says Yazhuo Zhang, an Emory Ph.D. student and co-first author of the paper. "It is already performing well but we are getting a lot of good suggestions to make it even better. That's the beauty of the open-source world."

Zhang shares first authorship of the paper with Juncheng (Jason) Yang, who received his master's degree in computer science at Emory and is now a Ph.D. candidate at Carnegie Mellon.

"SIEVE is an easy improvement of a tried-and-true cache-eviction algorithm that's been in use for decades—which is literally like centuries in the world of computing," says Ymir Vigfusson, associate professor in Emory's Department of Computer Science.

Vigfusson is co-senior author of the paper, along with Rashmi Vinayak, an associate professor in Carnegie Mellon's computer science department. Yao Yue, a computer engineer at the Pelikan Foundation, is also a co-author.

In addition to its speed and effectiveness, a key factor sparking interest in SIEVE is its simplicity, lending it scalability.

"Simplicity is the ultimate sophistication," Vigfusson says. "The simpler the pieces are within a system designed to serve billions of people within a fraction of a second, the easier it is to efficiently implement and maintain that system."

Keeping 'hot objects' handy

Many people understand the value of regularly reorganizing their clothing closet. Items that are never used can be tossed and those that are rarely used can be moved to the attic or some other remote location. That leaves the items most commonly worn within easy reach so they can be found quickly, without rummaging around.

A cache is like a well-organized closet for computer data. The cache is filled with copies of the most popular objects requested by users, or "hot objects" in IT terminology. The cache maintains this small collection of hot objects separately from a computer network's main database, which is like a vast warehouse filled with all the information that could be served by the system.

Caching hot objects allows a networked system to run more efficiently, rapidly responding to requests from users. A web application can effectively handle more traffic by popping into a handy closet to grab most of the objects users want rather than traveling down to the warehouse and searching through a massive database for each request.

"Caching is everywhere," Zhang says. "It's important to every company, big or small, that is using web applications. Every website needs a cache system."

And yet, caching is relatively understudied in the computer science field.

How caching works

While caching can be thought of as a well-organized closet for a computer, it is difficult to know what should go into that closet when millions of people, with constantly changing needs, are using it.

The fast memory of the cache is expensive to run yet critical to a good experience for web users. The goal is to keep the most useful future information within the cache. Other objects must be continuously winnowed out, or "evicted" in tech terminology, to make room for the changing array of hot objects.

Cache-eviction algorithms determine what objects to toss and when to do so.

FIFO, or "first-in, first-out," is a classic eviction algorithm developed in the 1960s. Imagine objects lined up on a conveyor belt. Newly requested objects enter on the left and the oldest objects get evicted when they reach the end of the line on the right.

In the LRU ("least recently used") algorithm, the objects also move along the line towards eviction at the end. However, if an object is requested again while it moves down the conveyor belt, it gets moved back to the head of the line.

Hundreds of variations of eviction algorithms exist but they have tended to take on greater complexity to gain efficiency. That generally means they are opaque to reason about and require high maintenance, especially when dealing with massive workloads.

"If an algorithm is very complicated, it tends to have more bugs, and all of those bugs need to be fixed," Zhang explains.

A simple idea

Like LRU and some other algorithms, SIEVE makes a simple tweak on the basic FIFO scheme.

SIEVE initially labels a requested object as a "zero." If the object is requested again as it moves down the belt, its status changes to "one." When an object labeled "one" makes it to the end of the line it is automatically reset to "zero" and evicted.

A pointer, or "moving hand," also scans the objects as they travel down the line. The pointer starts at the end of the line and then jumps to the head, moving in a continuous circle. Anytime the pointer hits an object labeled "zero," the object is evicted.

"It's important to evict unpopular objects as quickly as possible, and SIEVE is very fast at this task," Zhang says.

In addition to this quick demotion of objects, SIEVE manages to maintain popular objects in the cache with minimal computational effort, known as "lazy promotion" in computer terminology. The researchers believe that SIEVE is the simplest cache-eviction algorithm to effectively achieve both quick demotion and lazy promotion.

A lower miss ratio

The purpose of caching is to achieve a low miss ratio—the fraction of requested objects that must be fetched from "the warehouse."

To evaluate SIEVE, the researchers conducted experiments on open-source web-cache traces from Meta, Wikimedia, X and four other large datasets. The results showed that SIEVE achieves a lower miss ratio than nine state-of-the-art algorithms on more than 45% of the traces. The next best algorithm has a lower miss ratio on only 15%.

The ease and simplicity of SIEVE raises the question of why no one came up with the method before. The SIEVE team's focus on how patterns of web traffic have changed in recent years may have made the difference, Zhang theorizes.

"For example," she says, "new items now become 'hot' quickly but also disappear quickly. People continuously lose interest in things because new things keep coming up."

Web-cache workloads tend to follow what are known as generalized Zipfian distributions, where a small subset of objects account for a large proportion of requests. SIEVE may have hit a Zipfian sweet spot for current workloads.

"It is clearly a transformative moment for our understanding of web-cache eviction," Vigfusson says. "It changes a construct that's been used blindly for so long."

Even a tiny improvement in a web-caching system, he adds, can save millions of dollars at a major data center.

Zhang and Yang are on track to receive their Ph.D.s in May.

"They are doing incredible work," Vigfusson says. "It's safe to say that both of them are now among the world experts on web-cache eviction."

More information: Yazhuo Zhang et al, SIEVE is Simpler than LRU: an Efficient Turn-Key Eviction Algorithm for Web Caches

Provided by Emory University