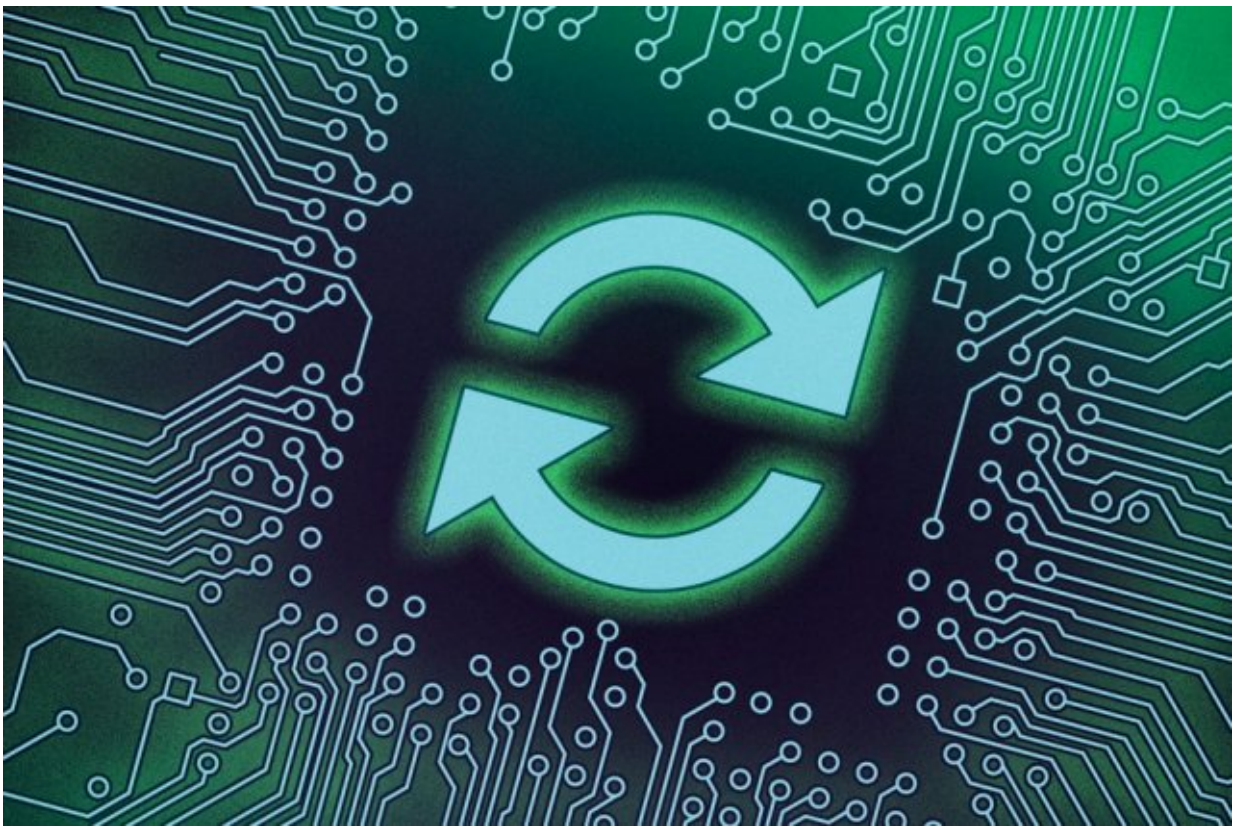
Scheme would make new high-capacity data caches 33 to 50 percent more efficient

October 23 2017, by Larry Hardesty



Credit: Massachusetts Institute of Technology

In a traditional computer, a microprocessor is mounted on a "package," a small circuit board with a grid of electrical leads on its bottom. The package snaps into the computer's motherboard, and data travels

between the processor and the computer's main memory bank through the leads.

As processors' transistor counts have gone up, the relatively slow connection between the processor and main memory has become the chief impediment to improving computers' performance. So, in the past few years, chip manufacturers have started putting dynamic random-access memory—or DRAM, the type of memory traditionally used for main memory—right on the chip package.

The natural way to use that memory is as a high-capacity cache, a fast, local store of frequently used data. But DRAM is fundamentally different from the type of memory typically used for on-chip caches, and existing cache-management schemes don't use it efficiently.

At the recent IEEE/ACM International Symposium on Microarchitecture, researchers from MIT, Intel, and ETH Zurich presented a new cache-management scheme that improves the data rate of in-package DRAM caches by 33 to 50 percent.

"The bandwidth in this in-package DRAM can be five times higher than off-package DRAM," says Xiangyao Yu, a postdoc in MIT's Computer Science and Artificial Intelligence Laboratory and first author on the new paper. "But it turns out that previous schemes spend too much traffic accessing metadata or moving data between in- and off-package DRAM, not really accessing data, and they waste a lot of bandwidth. The performance is not the best you can get from this new technology."

Cache hash

By "metadata," Yu means data that describe where data in the cache comes from. In a modern computer chip, when a processor needs a particular chunk of data, it will check its local caches to see if the data is

already there. Data in the caches is "tagged" with the addresses in main memory from which it is drawn; the tags are the metadata.

A typical on-chip cache might have room enough for 64,000 data items with 64,000 tags. Obviously, a processor doesn't want to search all 64,000 entries for the one that it's interested in. So cache systems usually organize data using something called a "hash table." When a processor seeks data with a particular tag, it first feeds the tag to a hash function, which processes it in a prescribed way to produce a new number. That number designates a slot in a table of data, which is where the processor looks for the item it's interested in.

The point of a hash function is that very similar inputs produce very different outputs. That way, if a processor is relying heavily on data from a narrow range of addresses—if, for instance, it's performing a complicated operation on one section of a large image—that data is spaced out across the cache so as not to cause a logjam at a single location.

Hash functions can, however, produce the same output for different inputs, which is all the more likely if they have to handle a wide range of possible inputs, as caching schemes do. So a cache's hash table will often store two or three data items under the same hash index. Searching two or three items for a given tag, however, is much better than searching 64,000.

Dumb memory

Here's where the difference between DRAM and SRAM, the technology used in standard caches, comes in. For every bit of data it stores, SRAM uses six transistors. DRAM uses one, which means that it's much more space-efficient. But SRAM has some built-in processing capacity, and DRAM doesn't. If a processor wants to search an SRAM cache for a data

item, it sends the tag to the cache. The SRAM circuit itself compares the tag to those of the items stored at the corresponding hash location and, if it gets a match, returns the associated data.

DRAM, by contrast, can't do anything but transmit requested data. So the processor would request the first tag stored at a given hash location and, if it's a match, send a second request for the associated data. If it's not a match, it will request the second stored tag, and if that's not a match, the third, and so on, until it either finds the data it wants or gives up and goes to main memory.

In-package DRAM may have a lot of bandwidth, but this process squanders it. Yu and his colleagues—Srinivas Devadas, the Edwin Sibley Webster Professor of Electrical Engineering and Computer Science at MIT; Christopher Hughes and Nadathur Satish of Intel; and Onur Mutlu of ETH Zurich—avoid all that metadata transfer with a slight modification of a memory management system found in most modern chips.

Any program running on a computer chip has to manage its own memory use, and it's generally handy to let the program act as if it has its own dedicated memory store. But in fact, multiple programs are usually running on the same chip at once, and they're all sending data to main memory at the same time. So each core, or processing unit, in a chip usually has a table that maps the virtual addresses used by individual programs to the actual addresses of data stored in main memory.

Yu and his colleagues' new system, dubbed Banshee, adds three bits of data to each entry in the table. One bit indicates whether the data at that virtual address can be found in the DRAM cache, and the other two indicate its location relative to any other data items with the same hash index.

"In the entry, you need to have the physical address, you need to have the virtual address, and you have some other data," Yu says. "That's already almost 100 bits. So three extra bits is a pretty small overhead."

There's one problem with this approach that Banshee also has to address. If one of a chip's cores pulls a data item into the DRAM cache, the other cores won't know about it. Sending messages to all of a chip's cores every time any one of them updates the cache consumes a good deal of time and bandwidth. So Banshee introduces another small circuit, called a tag buffer, where any given core can record the new location of a data item it caches.

Any request sent to either the DRAM cache or main memory by any core first passes through the tag buffer, which checks to see whether the requested tag is one whose location has been remapped. Only when the buffer fills up does Banshee notify all the chips' cores that they need to update their virtual-memory tables. Then it clears the buffer and starts over.

The buffer is small, only 5 kilobytes, so its addition would not use up too much valuable on-chip real estate. And the researchers' simulations show that the time required for one additional address lookup per memory access is trivial compared to the bandwidth savings Banshee affords.

More information: Xiangyao Yu et al. Banshee, *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture - MICRO-50 '17* (2017). DOI: 10.1145/3123939.3124555

This story is republished courtesy of MIT News (web.mit.edu/newsoffice/), a popular site that covers news about MIT research, innovation and teaching.

TechXplore

Provided by Massachusetts Institute of Technology

Citation: Scheme would make new high-capacity data caches 33 to 50 percent more efficient (2017, October 23) retrieved 6 May 2024 from https://techxplore.com/news/2017-10-scheme-high-capacity-caches-percent-efficient.html

This document is subject to copyright. Apart from any fair dealing for the purpose of private study or research, no part may be reproduced without the written permission. The content is provided for information purposes only.