'Hiding' network latency for fast memory in data centers

July 29 2020, by Steve Crang

Connecting memory spread across a data center via networking can drastically improve job speed and the center's efficiency—but the performance cost of accessing memory over networks was too great for widespread adoption. A new tool called Leap changes that. Credit: CSE

Sharing server memory between applications in large computer clusters is still a major goal for cloud and high-performance computing communities. Through fast networking technology, the memory available throughout the center's server racks could be managed by schedulers as though it were a single resource, providing a major boost to speed and performance.

A service developed by University of Michigan researchers called Infiniswap made this technology—called "memory disaggregation"—feasible in 2017, but it still suffered from several latency overheads that made real-world adoption unlikely. Now, a new system from the same lab called Leap improves upon this and other disaggregation solutions by applying a technique called prefetching to remote memory environments. Leap earned a Best Paper Award at the 2020 USENIX Annual Technical Conference.

The main problem facing practical memory disaggregation, according to Leap co-author Hasan Al Maruf, was a speed difference between local and remote memory access.

"When you access memory locally," says Maruf, a Ph.D. student in U-M's Computer Science and Engineering division, "memory runs at the nanosecond level. But when you go to the network, things run at the microsecond level."

It takes an average of four or five microseconds to load a page of data from memory remotely (a page is the smallest unit of data the processor typically fetches from memory at a time). In the worst case it can take up to 40-50 microseconds.

This might not sound like much, but the order of magnitude delay in moving from nanoseconds to microseconds represents a major performance drop—about the same as comparing RAM with a solid state drive, or solid state with a hard disk drive. The tradeoff was too great, and existing methods proved favorable to remote memory access.

"Our measurements show that an average 4KB remote page access takes close to 40 microseconds in state-of-the-art memory disaggregation systems like Infiniswap," write Maruf and co-author Prof. Mosharaf Chowdhury. "Such high access latency significantly affects performance

because memory-intensive applications can tolerate at most single microsecond latency."

This issue was compounded further by overheads introduced by the operating system. The data path used by operating systems in these remote settings to access an address in memory was originally designed for interactions with local disks, which operate at the even slower millisecond level. These data paths come bundled with features meant to hide this disk access time that aren't practical for data center networks and end up bogging everything down.

It was Maruf and Chowdhury's goal with Leap to "hide" these sources of latency with two tricks: prefetching pages wherever possible, and using more efficient data paths that allow them to discard the operating system's irrelevant disk-access features.

Their goal with the prefetcher is to reduce the number of calls to remote memory in the critical path needed by the program, keeping useful data close at hand on the local machine running the program. This technique is now commonplace in single-machine settings where the most common performance bottleneck is making calls to the hard disk. There, systems try to smartly identify additional instructions in a sequence that might be useful soon and pull them into the processor's much-faster cache.

Emulating this technique becomes difficult in remote memory settings because related data might not be stored sequentially—in fact, it might be stored randomly all across the data center's available memory. Existing attempts to solve this problem identify patterns in how memory is accessed in real time, or keep track of the program's memory access footprint on the entire virtual memory address space.

Both of these techniques require a lot of CPU usage and additional memory overhead.

Instead, the authors only approximate patterns in a program's memory access. This allows Leap to identify the most clear opportunities to prefetch additional data without having to take up resources being more precise—only those cases where the program accesses the same addresses in memory repeatedly, the so-called majority access pattern, will be identified.

Combined with using simplified data paths that are recorded for each address the program accesses in remote memory, the prefetcher allows nearly all applications to run as if they were working with local memory.

"This prefetching solution helps to hide the network latency, and the data path makes sure the operating system has no overhead," Maruf says.

Using Leap to fix the system's data paths provided single-microsecond latency at worst in 95% of tasks, with an average of five or six microseconds. Including the prefetcher in their tests, Leap provided sub-microsecond latency, or latency in nanoseconds, on 85% of tasks.

These speeds provide a big boost to the future of memory disaggregation in data centers in the long term. In the short term, Leap can provide latency benefits when reading pages from storage devices other than remote memory, like traditional disks and solid state drives.

"Making something exist and making things practical are two different things," he says. "If we didn't cross the boundary of this performance latency overhead, whatever system we have wouldn't be put to use."

But Maruf says there are still barriers to its practical adoption. Chowdhury's lab is tackling security and resiliency challenges facing the field in future work, which themselves may introduce additional latency in the systems.

"We are exploring in many different directions," says Maruf, "and if we can finish them all we hope we'll make things more practical and usable in public cloud setups."

Leap was presented in the paper "Effectively Prefetching Remote Memory with Leap" at the 2020 USENIX Annual Technical Conference, held remotely.

More information: Effectively Prefetching Remote Memory with Leap. www.usenix.org/system/files/atc20-maruf.pdf

Provided by University of Michigan