BREAKING THE MULTICORE BOTTLENECK

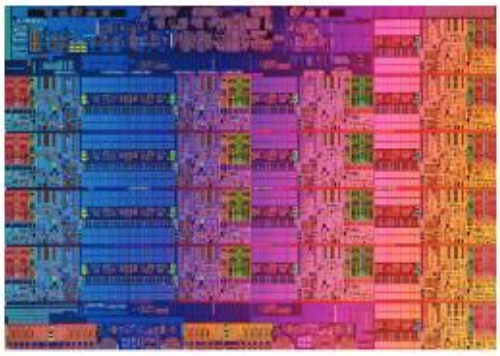
Simple hardware speeds core-to-core communication

Engineers at North Carolina State University and at Intel have come up with a solution to one of the modern microprocessor's most persistent problems: communication among the processor's many cores. Their answer is a dedicated set of logic circuits they call the Queue Management Device, or QMD. In simulations, integratingthe QMD with the processor's on-chip network at a minimum doubled core-to core communication speed and, in some cases, boosted it much further. Even better, **as the number of cores was increased, the speedup became more pronounced.**

In the last decade, microprocessor designers started putting multiple copies of processor cores ona single die as a way to continue the rate of performance improvement computer makers had enjoyed without causing chip-killing hot spots to form on the CPU. But that solution comes with complications. For one, it means that software programs have to be written so that work is divided among processor cores. The result: Sometimes different cores need to work on the same data or must coordinate the passing of datafrom one core to another.

To prevent the cores from wantonly overwriting one another's information, processing data out of order, or committing other errors, multicore processors use lock-protected software queues. These are data structures that coordinate the movement of and access to information according to software-defined rules. But all that extra software comes with significant overhead, which only gets worse as the number of cores increases. "Communications between cores is becoming a bottleneck," says Yan Solihin, a professor of electrical and computer engineering who led the work at NC State, in Raleigh.

Thesolution-born of a discussion with Intel engineers and executed by Solihin's student, Yipeng Wang, at NC State and at Intel-was to turn the software queue into hardware. This effectively turned three multistep software-queue operations into three simple instructions: Add data to the queue, take data from the queue, and put data close to where it's going to be needed next. Compared with just using the software solution, the QMD sped up a sample task such as



IT'SGETTING CROWDED: This Intel Haswell EXXeon E7 V3 processor has 18 cores trying to work together without messing up one another's calculations. A bit of additional hardware could speed up communication among the cores.

packet processing - **like network nodes do on the Internet-by a greater and greater amount the more cores were involved.** For16 cores, QMD worked 20 times as fast as the software could.

Once they achieved this result, the engineers reasoned that the QMD might be able to do a few other tricks-such as turning more software into hardware. They added more logic to the QMD and found it could speed up several other core-communications-dependent functions, including MapReduce, a technology Google pioneered for distributing work to different cores and collecting the results.

Srini Devadas, an expert in cache control systems at MIT, says the QMD addresses "a very important problem." Devadas's own solution for the use of caches by multiple cores-or even multiple processors-is more radical than the QMD. Called Tardis, it's a complete rewrite of the cache management rules, and so it is a solution aimed at processors and systems of processors further in the future. But QMD, Devadas says, has nearer-term potential. "It's the kind of work that would motivate Intel-putting in a small piece of hardware for a significant improvement."

The Intel engineers involved couldn't comment on whether QMDwould find its way into future processors. However, they are actively researching its potential. (Wang is nowa research scientist at Intel.) The engineers hope that QMD, among other extensions of the concept, can simplify communication among the cores and the CPU's input/output system.

Solihin, meanwhile, is inventing other types of hardware accelerators. "We have to improve performance by improving energy efficiency. The only way to dothat is to move some software to hardware. The challenge is to figure out which software is used frequently enough that wecould justify implementing it in hardware," he says. "There is a sweet spot." -SAMUEL K. MOORE

This is NOT better than the Ideal Case but *may* be an improvement on the upper bound of Amdahl's Law

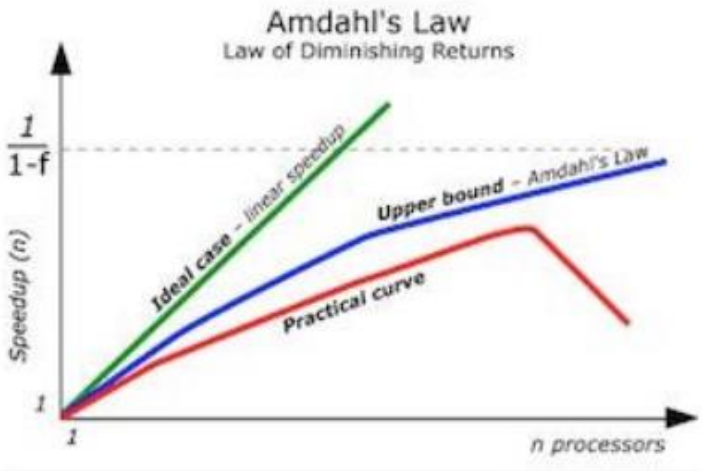


Image from: https://www.javacodegeeks.com/wp-content/uploads/2