**New System Could Break Bottleneck in Microprocessors**

By [Samuel K. Moore](http://spectrum.ieee.org/author/moore-samuel-k) <http://spectrum.ieee.org/> Posted 12 Sep 2016

Intel and other multicore processor makers want cores to be able to communicate with each other faster. More cores (the Haswell-EX Xeon E7-8890 V3 shown here has 18) typically means much more time coordinating communications.

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 between the processor’s many cores. Their answer is a dedicated set of logic circuits they call the Queue Management Device, or QMD. In simulations, integrating the QMD with the processor’s on-chip network, at a minimum, doubled core-to-core communication speed, and in some cases, boosted it much farther. Even better, as the number of cores was increased, the speed-up became more pronounced.

In the last decade, microprocessor designers started putting [multiple copies of processor cores on a single die](http://spectrum.ieee.org/semiconductors/processors/multicore-cpu-processor-proliferation) as a way to continue the rate of performance improvement computer makers had enjoyed without chip-killing hot spots forming on the CPU. But that solution comes with [complications](http://spectrum.ieee.org/computing/software/the-trouble-with-multicore). For one, it meant that software programs had to be written so that work was divided among processor cores. The result: Sometimes different cores would need to work on the same data or have to coordinate the passing of data from one core to another.

To prevent the cores from wantonly overwriting each other’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](http://www.ece.ncsu.edu/people/solihin), a professor of electrical and computer engineering who led the work at NC State.

The solution—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 near where it’s going to be needed next. Compared with just using the software solution, the QMD sped up a sample task such as packet processing—like network nodes do on the Internet—by a greater and greater amount the more cores were involved. For 16 cores, QMD worked 20 times as fast as the software could.

Once they realized this result, the engineers reasoned that the QMD might be able to do a few other tricks—turning more software into hardware. They added more logic to the QMD and found that 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.

They aren’t done yet. “The next step is to figure out other types of hardware accelerators that would be useful,” says Solihin. “We have to improve performance by improving energy efficiency. The only way to do that is to move some software to hardware. The challenge is to figure out which software is used frequently enough that we could justify implementing it in hardware. There is a sweet spot,” he says.

Intel engineer Ren Wang is presenting the QMD speed-up results at the [25th Annual Conference on Parallel Architectures and Compilation Techniques](http://pactconf.org/), in Haifa, Israel this week.

**Top Programming Languages Trends: The Rise of Big Data**

**Languages like Go, Julia, R, Scala, and even Python are riding the number-crunching wave**

By Nicholas Diakopoulos <http://spectrum.ieee.org/> Posted 26 Jul 2016

In this article I’m going to focus on so-called big-data languages, such as [Julia](http://julialang.org/), [Python](https://www.python.org/), [R](https://www.r-project.org/), and [Scala](http://www.scala-lang.org/). Most of these are purpose-built for handling large amounts of numeric data, with stables of packages that can be tapped for quick big-data analytic prototyping. These languages are increasingly important, as they facilitate the mining of the huge data sets that are now routinely collected across practically all sectors of government, science, and commerce.

The biggest mover in this category was [Go](https://golang.org/), an open source language created by Google to help solve the company’s issues with scaling systems and concurrent programming back in 2007. In the [default *Spectrum* ranking](http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2016), it’s moved up 10 positions since 2014 to settle into 10th place this year. Other big-data languages that saw moves since 2014 in the *Spectrum* ranking were R and Scala, with R ascending 4 spots and Scala moving up 2 (although down from 2015, when it was up 4 places from its 2014 position). Julia was added to the list of languages we track in 2015, and in the past year it’s moved from rank 40 to 33, still a marginal player but clearly possessing some momentum in its growth.

The chief reason for Go’s quick rise in our ranking is the large increase in related activity on the [GitHub](https://github.com/) source code archive. Since 2014, the total number of repositories on GitHub that list Go as the primary language went up by a factor of more than four. If we look at just *active* GitHub repositories, then there are almost five times as many. There’s also a fair bit more chatter about the language on [Reddit](https://www.reddit.com/), with our data showing a threefold increase in the number of posts on that site mentioning the language.

Another language that has continued to move up the rankings since 2014 is R, now in fifth place. R has been lifted in our rankings by racking up more questions on [Stack Overflow](http://stackoverflow.com/)—about 46 percent more since 2014. But even more important to R’s rise is that it is increasingly mentioned in scholarly research papers. The *Spectrum* default ranking is heavily weighted toward data from [IEEE Xplore](http://ieeexplore.ieee.org/Xplore/home.jsp), which indexes millions of scholarly articles, standards, and books in the IEEE database. In our 2015 ranking there were a mere 39 papers talking about the language, whereas this year we logged 244 papers.

Contrary to the substantial gains in the rankings seen by open source languages such as Go, Julia, R, and Scala, proprietary data-analysis languages such as [Matlab](http://www.mathworks.com/products/matlab/) and [SAS](http://www.sascommunity.org/wiki/Main_Page) have seen a drop-off: Matlab fell four places in the rankings since 2014 and SAS has fallen seven. However, it’s important to note that both of those languages are still growing; it’s just that they’re not growing as fast as some of the languages that are displacing them.

When we weight the rankings toward jobs, we continue to see heavily used languages like [Java](https://java.com/) and Python dominate. But recruiters are much more interested in R and Scala in 2016 then they were in 2014. When we collected data in 2014, there were only 136 jobs listed for Scala on CareerBuilder and Dice. But by 2016 there was more than a fourfold increase, to 631 jobs.

This growth invites the question [whether R can ever unseat Python](http://www.kdnuggets.com/2015/05/r-vs-python-data-science.html) or Java as the top languages for big data. But while R has seen huge gains over the last few years, Python and Java really are 800-pound gorillas. For instance, we found roughly 15 times as many job listings for pythonistas as for R developers. And while we measured about 63,000 new GitHub repositories in the last year for R, there were close to 458,000 for Python. Although R may be great for visualization and exploratory analysis and is clearly popular with academics writing research papers, Python has significant advantages for users in production environments: It’s more easily integrated into production data pipelines, and as a general purpose language it simply has a broader array of uses.

These data illustrate that despite the desire of some coders to evaluate languages on purely internal merits—the elegance of their syntax, or the degree and nature of the abstractions used—a big driver for a language’s popularity will always be the domains that it targets, either by design or through the availability of supporting libraries.

**Use a GPU to Turn a PC Into a Supercomputer**

By Mark Anderson <http://spectrum.ieee.org/> Posted 8 Jul 2016

As Moore’s Law [slows](http://spectrum.ieee.org/computing/hardware/moores-law-might-be-slowing-down-but-not-energy-efficiency) for CPUs, dedicated graphics co-processors are picking up some of the slack. Just as GPUs are changing the game in [deep learning](http://spectrum.ieee.org/tech-talk/semiconductors/processors/a-deep-learning-ai-chip-for-your-phone) and [autonomous cars](http://spectrum.ieee.org/cars-that-think/transportation/self-driving/nvidia-wants-to-build-the-robocars-brain), the GPU-powered desktop PC might even begin to keep pace with the conventional supercomputer for a portion of supercomputer applications.

For instance, a group of Russian scientists are reporting this month that they’ve been able to solve computational problems in nuclear physics using an off-the-shelf, high-end PC containing a GPU. And, they say, after fine-tuning their algorithm for GPUs, they were able to run their calculations faster than the traditional, CPU-powered supercomputer their colleagues use. Bonus: they ran those calculations for free as opposed to their colleagues, who must pay for access to the supercomputer to run their computations.

“I have no doubts that many research groups over the world can reach the similar results in their own fields such as geophysics, seismology, plasma physics, [medical] diagnostics, etc.” says Vladimir Kukulin, professor of theoretical physics at [Lomonosov Moscow State University](http://www.msu.ru/en/). “But only by combining two above ingredients: Reformulation of the whole problem, and then by inventing some effective way how to parallelize the whole execution in thousands or even millions of independent threads.”

The problem Kukulin’s group was tackling involved the extensive calculations needed to describe scattering problems in their field — such as when one nucleon collides with a particle or another nucleon and produces a spray of particles and daughter nuclei as a result. This nuclear many-body problem, Kukulin says, could require calculations involving matrices containing millions of elements.

Matrix algebra with this many moving parts can stymie even a supercomputer. But, Kukulin says, his group realized, too, that calculations with giant matrices can mean independent threads of instructions to run in parallel with many other similar threads. The parallelizability of his group’s nuclear calculations, in other words, meant it was a prime candidate for running efficiently on a GPU.

The GPU, [originally designed](https://en.wikipedia.org/wiki/Graphics_processing_unit#History) to handle matrix-heavy calculations needed to generate real-time graphics, is finding unexpected applications in a number of fields today, including [Bitcoin mining](http://spectrum.ieee.org/computing/networks/bitcoins-computing-crisis), [molecular modeling](https://en.wikipedia.org/wiki/Molecular_modeling_on_GPUs), and the applications noted above. Kukulin says the list of computational tasks the GPU can handle, a list that now includes nuclear physics, will only increase.

Overall, he says the kinds of problems that could lend themselves to GPU-supercomputing on the cheap are those whose individual elements are not interdependent on one another. Because interdependence means that individual elements (i.e. threads in a GPU’s calculations) would have to go through regular if-then logic gates, checking for each element’s ongoing influence on other elements of the calculation.

And such conditional logic steps are probably going to involve the CPU, which slows an otherwise streamlined calculation down. Instead, to maximize the speedup a GPU can produce, he says it’s best to find a way of expressing one’s problem — or find an [approximation](https://en.wikipedia.org/wiki/Finite_difference_method) that enables the expression — as a system containing many discrete and unconnected elements.

“You should write your problem into a form that allows you to massively parallelize,” he says. “It’s necessary to avoid somehow any conditional.”

So complicated simulations, where every component’s interaction is dependent on other components and their trajectories in time, would be difficult to translate into a GPU-ready problem. By contrast, he says, tsunami[early warning systems](https://www.researchgate.net/publication/277757438_Tsunami-HySEA_A_GPU-based_model_for_Tsunami_Early_Warning_Systems) that predict tidal wave landfalls in faster-than-real-time have sped up when run on GPUs as opposed to traditional CPU supercomputers.

Kukulin says 3D ultrasound imaging, a compute-intensive medical diagnostic tool, could be more widely embraced if medical offices needed only a few thousand dollars for a desktop PC as opposed to many thousands of dollars for access to a supercomputer.

Ultimately, then, problems fit for GPU speedup require not just programming finesse but also expertise in the field of application to find the right way through the problem that could produce a GPU speedup.

“This is some art, but not [just] in programming,” Kukulin says.

Kukulin’s group’s research was [published](http://www.sciencedirect.com/science/article/pii/S0010465516300765) in the July issue of the journal*Computer Physics Communications* (and is also [available](http://arxiv.org/abs/1508.07441) on the arXiv preprint server).

**Tesla's Massive New Autopilot Update Is Released, Promising Safer Driving**

By [Philip E. Ross](http://spectrum.ieee.org/author/ross-philip-e) <http://spectrum.ieee.org/> Posted 22 Sep 2016

A long-heralded update to Tesla Motor’s Autopilot has just been made available for download. First reports suggest that it’s as big a change in the semiautonomous driving system as Tesla CEO [Elon Musk](https://twitter.com/elonmusk)had promised.

One key element of the upgrade is making more use of the car’s existing radar capabilities, both to perceive the road in real time and to map it so that subsequent Tesla cars can distinguish earlier fixed features from new, perhaps threatening ones. Another key element is saving drivers from over-dependence on the software.

Either of those points might have saved the Tesla owner [who died last May](http://spectrum.ieee.org/cars-that-think/transportation/self-driving/tesla-autopilot-crash-why-we-should-worry-about-a-single-death) when his Autopilot, apparently unsupervised by the driver, drove into the side of a tractor trailer. That is the first fatality known to have been caused by a modern robotic driving system.

“We believe it would have seen a large metal object across the road,” [Musk said](http://www.bloomberg.com/news/articles/2016-09-11/tesla-alters-autopilot-to-emphasize-radar-images-over-cameras) in a conference call earlier this month, referring to the trailer. “Knowing that there is no overhead road sign there, it would have braked.”

(Another Tesla driver died in China in January, in a case [now under litigation](http://www.scmp.com/business/companies/article/2021356/father-chinese-man-killed-tesla-crash-files-lawsuit-over)there, but it isn’t clear whether the Autopilot was operating at the time of the crash.)

Tesla’s preference for radar over lidar, the laser-ranging equivalent, makes the company a little unusual in autoland. Lidar has far better resolution—unlike radar it can see road markings and make out the shapes of signs and other things even at a distance.

Radar, however, is cheaper, more compact, and far better at seeing through rain and snow. And Tesla needs this immediate practicality because it’s incrementally raising the capability of its cars’ “advanced driver assistance systems,” or ADAS, to a fully self-driving level. By contrast, [Google](http://spectrum.ieee.org/image/MjUwNTkyNA.jpeg), [Ford](http://spectrum.ieee.org/image/MjgwOTUxMQ.jpg) and[Uber](http://spectrum.ieee.org/image/MjgwMjI0NQ.jpg) are aiming to produce a fully robotic car in one fell swoop. They now festoon their experimental cars with lidar in the expectation that it will become cheaper, smaller and more capable by the time that car is ready, five years (at least) from now.

Tesla’s Autopilot 8.0 goes further than ever to keep the driver’s eyes on the road. For instance, it will sound the alarm if your hand’s off the wheel, then does it with increasing insistance until, after the third time, the Autopilot will disengage for the remainder of the trip.

How far these changes will go to prevent accidents, small and large, remains to be seen. For now, though, the select reviewers who have been beta-testing the car say that it certainly drives in less machine-like way.

“It’s only human to want to give the truck a little more space and hug the outer edge of the lane,” [writes Tom Randall](http://www.bloomberg.com/news/features/2016-09-22/tesla-drivers-wake-up-to-a-serious-upgrade) in *Bloomberg News*. “With the upgrade, the car is beginning to act a little more human, adjusting its position in the lane to account for perceived threats from the sides.”

One ho-hum aspect of today’s upgrade would once have been the most striking thing of all: it’s all done through an over-the-air download. Tesla [pioneered](https://forums.tesla.com/it_IT/forum/forums/first-overtheair-firmware-update) this trick, and now other automakers are [following suit](http://www.autonews.com/article/20160125/OEM06/301259980/over-the-air-updates-on-varied-paths). Here Tesla has a built-in advantage over other car makers: it sells cars direct to the public, so upgrades can go straight to the customer without alienating a dealer network.