OpenACC Starts to Gather Developer Mindshare

May 17, 2012 | Michael Feldman

PGI, Cray, and CAPS enterprise are moving quickly to get their new OpenACC-supported compilers into the hands of GPGPU developers. At NVIDIA's GPU Technology Conference (GTC) this week, there was plenty of discussion around the new HPC accelerator framework, and all three OpenACC compiler makers, as well as NVIDIA, were talking up the technology.

Announced at the Supercomputing Conference (SC11) last November, OpenACC is an open standard API developed by NVIDIA, PGI, Cray, and CAPS, to provide a high-level programming framework for programming accelerators like GPUs. OpenACC uses compiler directives, which programmers insert into high-level source (e.g., C, C++ or Fortran), to tell the compiler to execute specific pieces of the code on the accelerator hardware.

GTC conference-goers had plenty of opportunity to encounter OpenACC this week. There two OpenACC tutorials for would-be developers, one by NVIDIA, and the other by CAPS enterprise. In addition, there were four other sessions hosted by Cray, CAPS, and PGI throughout the week. That's not counting the numerous mentions OpenACC got during other presentations involving GPGPU programming.

The technology is still in its infancy though. The PGI and Cray compilers are pre-production versions. CAPS first commercial offering is just two weeks old.

The initial goal of OpenACC is to bring more developers (and codes) into GPU computing, especially those not being served by the lower-level programming frameworks like CUDA and OpenCL. While CUDA is widely used in universities and in the technical computing realm, and OpenCL is emerging as an open standard for parallel computing, neither is particular attractive to commercial developers.

Most programmers are used to writing high-level code that focuses on the problem at hand, without have to worry about the vagaries of the underlying hardware. That hardware independence is also what makes OpenACC attractive for codes that need to span different processor architectures.

That assumes, of course, that compiler will support multiple accelerator chips. The first crop of OpenACC-enabled compilers from PGI, CAPS and Cray only generate code for NVIDIA GPUs -- not too surprising when you consider NVIDIA's current dominance in HPC acceleration. However all of the compiler efforts plan to widen the aperture of hardware support.

CAPS is perhaps most aggressive in this regard. According to CAPS CTO François Bodin, his company plans to add OpenACC support for AMD GPUs, x86 multicore CPUs and even the Tegra 3 microprocessor, an ARM-GPU design that will be used to power an experimental HPC clusters at the Barcelona Supercomputing Center (BSC). Bodin also said that they have an Intel MIC (Many Integrated Core) port of OpenACC in the pipeline. All of these compiler ports should be available later this year.

PGI is keeping its OpenACC development plans a little closer to the vest. But according to PGI compiler engineer Michael Wolfe, they have received requests for OpenACC support for nearly every processor and co-processor used in high performance computing. The compiler maker will undoubtedly be developing some of these over the next year.

Likewise for Cray, although its OpenACC compiler support is focused on the underlying accelerators of its own XK6 supercomputers. At this point, that's confined to NVIDIA GPUs. Cray (which also carries CAPS and PGI compilers for its customers) has a unique OpenACC offering in that it supports those directives in PGAS languages Co-Array Fortran and Unified Parallel C (UPC) on the XK6.

Besides its applicability to multiple hardware platforms, OpenACC is just plain easier to use when you have lots of existing code. For one thing, OpenACC lets you attack the acceleration in steps. CUDA and OpenCL ports usually require code rewrites of at least a sizeable chunk of the application being accelerated, using low-level APIs. With OpenACC, the programmer just has to insert high-level directives into existing source, and this can be done iteratively, gradually putting more and more of the code under OpenACC control. This, say, PGI's Wolfe, is "a hell of a lot more productive" than the low-level approach.

Even at the national labs and research centers, where there are computer scientists aplenty, OpenACC is starting to be recognized as an easier path to bring acceleration to hundreds of thousands of line of legacy codes. NASA Ames is already using PGI's compiler to speed up some of their CFD codes on one of their GPU clusters. And the upcoming deployments of multi-petaflop GPU-based supercomputers like "Titan" at Oak Ridge National Lab, should provide a lot more opportunities for OpenACC-based application development. Titan project director Buddy Bland is on record endorsing the technology for software development on that machine.

As with all parallel programming though, there's no free lunch to be had. In general, the programmer is probably going to sacrifice some runtime performance (compared to CUDA, for example) for the sake of programmer productivity. But there seems to be a general consensus that intelligent use of directives can easily get you to within 10 or 15 percent the performance of a low-level implementation. But as CAPS' Bodin explains, to get in that close, "you have to know what you're doing." On the other hand, as the compiler technology matures and developers get more adept with OpenACC, the performance gap could narrow even further.

The other problem is just a lack of accelerator diversity at the moment. With Intel MIC waiting in the wings, and AMD still pretty much a no-show with server-side GPUs, there's no immediate need to support anything but NVIDIA's GPU architecture right now. Worse, both Intel and AMD are backing other parallel computing frameworks that they are rolling into to their accelerator programs: OpenMP, Cilk Plus, and TBB for Intel; OpenCL and C++ AMP for AMD.

Fortunately, it probably doesn't matter that Intel and AMD haven't hopped on the OpenACC bandwagon. PGI and CAPS can still produce compilers targeting Intel MIC or AMD GPUs, or whatever else comes along. And as long as there are at least two compiler vendors offering such support, the community should be satisfied.

The end game, though, is to fold the OpenACC capabilities into OpenMP. If and when that happens, both Intel, AMD will throw their support behind it. OpenMP has been around for 15 years and is a true industry standard.

There is currently a Working Group on Accelerators in the OpenMP consortium, which is looking at incorporating accelerator directives into the next OpenMP release. And while those directives will be based on the OpenACC directives, they are not likely to be adopted as is. There's a real risk that if the process gets drawn out much longer and OpenACC captures a critical mass of users, there will end up being two directive-based accelerator standards to choose from.

Twas ever thus.

Related Articles

CAPS Entreprise Now Supports OpenACC Standard

OpenMP Announces Improvements for Multicore and Accelerators

OpenACC Support Available With New PGI Accelerator Fortran and C Compilers

NVIDIA Announces Initial Results of Directives-Based GPU Computing Program

NVIDIA Eyes Post-CUDA Era of GPU Computing
