OpenACC API

What is the OpenACC API?
OpenACC is a set of directive-based extensions to standard Fortran, C and C++ that enable offloading of data and compute-intensive loops and regions of code from a CPU host to an attached accelerator device.

As a high-level HPC parallel programming model, OpenACC is designed for performance on—and performance portability across—many types of platforms including GPUs, many-core and multi-core processors. OpenACC is complementary to existing HPC programming models including OpenMP, MPI, CUDA and OpenCL.

How does the OpenACC API work?
Based on directives supplied by the programmer in the form of Fortran comment statements and C/C++ pragmas, OpenACC compilers automatically map compute-intensive loops and manage data movement between the CPU host and the accelerator. In addition, the OpenACC API provides the programmer with directives to override the compiler’s mapping and data movement decisions when necessary. Using OpenACC, programmers can quickly determine if their code will benefit from acceleration. Because it is directive-based, OpenACC requires many fewer structural code changes than low-level accelerator programming models like CUDA or OpenCL. OpenACC directives are simply ignored by compilers not supporting the OpenACC API.

What challenge does the OpenACC API solve?
OpenACC is a high-level, performance—(performance-portable) portable programming model that helps developers to extend the performance of clustered MPI and shared-memory OpenMP-based applications to high-performance, energy-efficient accelerators running thousands or millions of lightweight threads.

What are the benefits of the OpenACC API?
OpenACC defines and enables a straightforward, higher-level path for developers interested in exploring the potential performance benefits of powerful many-core accelerators such as GPUs.

OpenACC preserves a common code base for both accelerated and non-accelerator enabled applications. OpenACC directives take the form of comments that are ignored by a compiler when targeting non-accelerator enabled systems. The specification was designed to encourage OpenACC implementations on multiple types and brands of accelerators (CPUs, APUs, GPUs). OpenACC application portability has been demonstrated across NVIDIA GPUs and AMD APUs/GPUs (PGI, CAPS) as well as with GPUs and the Intel Xeon Phi coprocessor (CAPS).

OpenACC provides an incremental path for moving legacy applications to accelerators. This means fewer structural changes to existing code are required than when using other approaches.

OpenACC establishes a baseline functional model for programming accelerators but at the same time provides programming tool vendors an opportunity to innovate by adding their own unique extensions.

Who can benefit from the OpenACC API?
Scientists, engineers and other domain experts interested in porting their applications to accelerators but who are not full-time software developers will find OpenACC a quick, straightforward, and performance-portable programming model.

Organizations with a significant investment in legacy production applications will find that OpenACC provides an incremental path for moving those applications to accelerators. It also may disturb the existing code less than other approaches.

ISVs interested in developing hardware-independent accelerated applications will find OpenACC to be a performance-portable approach for supporting multiple accelerator targets.

How did the OpenACC specification originate?
OpenACC was initially developed by PGI, Cray, and NVIDIA with support from CAPS, and is largely based on the PGI Accelerator programming model. Two versions of the specification have been ratified in its first three years, and the founders, along with several new member users, continue to aggressively push to refine and enhance the specification.

How does the OpenACC API relate to the OpenMP API?
The OpenMP API was adopted in the mid-1990s in response to efforts by multiple system vendors to define a standard for multi-threaded programming of symmetric multiprocessor (SMP) systems. At the time, systems had relatively few processing cores (two to 32 processors). With the development of CUDA and OpenCL, access to massively parallel many-core GPUs for general purpose computing became practical. Several high-level directive based models for these systems quickly followed. Much like SMP in the 1990s, users and vendors recognized the need for a directive-based many-core parallel programming standard. Cray, PGI and NVIDIA, along with CAPS, formed the OpenACC standards organization (openacc-standard.org) in 2010. Today, OpenACC and OpenMP are complements to one another much like OpenMP and MPI. The OpenACC API is defined, and has been demonstrated to be, interoperable with OpenMP.
Is the OpenACC API only for HPC?
No. The OpenACC API is for anyone interested in a quick and straightforward approach to programming accelerators, whether they are doing science in traditional HPC or working in sectors with similar needs such as Oil and Gas, Finance and Manufacturing.

Does the OpenACC API run on AMD GPUs and/or the Intel MIC?
Both CAPS and PGI support OpenACC on AMD GPUs. CAPS supports OpenACC on the Intel Xeon Phi coprocessor.

Does the OpenACC API run on top of OpenCL?
Yes. The CAPS compilers running on both AMD GPUs and Intel Xeon Phi coprocessors use those vendors’ OpenCL back-end. PGI’s OpenACC implementation on AMD GPUs does so as well.

Will AMD/Intel/Microsoft support OpenACC?
We can’t speak to the rate of external adoption or participation. We recommend you contact those other suppliers for details.

Will the OpenACC API run on NVIDIA GPUs with CUDA?
Yes. Programmers may wish to develop one part of their application using directives and call CUDA-accelerated libraries or routines in another. This interoperability is a key strength of OpenACC and allows programs to thoroughly optimize the very computationally intense portions of their code using CUDA C or CUDA Fortran.

Are compilers implementing the OpenACC API shipping today?
Yes. Cray has been shipping compilers with full support for OpenACC 1.0 since the release of Cray Compiler Environment (CCE) 8.1. Cray fully supports OpenACC 2.0 with the release of CCE 8.2. PGI has been supporting OpenACC 1.0 since the release of the 12.6 version of the PGI Accelerator Fortran and C compilers. Support for C++ has been available since version 13.6. CAPS implemented OpenACC 1.0 in version 3.1 of their CAPS compilers. Open source implementations for GNU and Open64 are also underway.

What is the current status of OpenACC 2.0?
The OpenACC 2.0 specification was ratified in June 2013. The version 2.0 specification clarifies some ambiguity in the 1.0 version, and adds several new features including support for function calls, unstructured data lifetimes, nested parallelism, support for multiple device types, atomic constructs and more.

When will the OpenACC vendors support the version 2.0 specification?
Cray provides full OpenACC 2.0 support, including separate compilation/linking and nested parallelism, in the Cray Compiler Environment (CCE) 8.2 release available now. PGI is rolling out OpenACC 2.0 features with their PGI 2014 release. Partial support for OpenACC 2.0 is already available in the latest versions of CAPS compilers with full support scheduled in December 2013.

Where can I learn more about the OpenACC API?
You can download the specifications and the reference card from the OpenACC website at www.openacc.org.

In addition, you can review presentations recorded from conferences such as Supercomputing, ISC, and the GPU Technology Conference.

How can I stay informed about the OpenACC organization?
Please sign up for OpenACC bulletins, and/or register to join the site to contribute to the OpenACC forums.

Both are available through links on the OpenACC website at www.openacc.org.

How can I join OpenACC.org?
Those parties interested in participating in the definition and direction of OpenACC are encouraged to join OpenACC.org. Three membership levels are available including member, supporter and a special low-cost membership for academic institutions. Find out more on the OpenACC website at www.openacc.org/join.