

What is OpenACC?

10/21/16 | A. Severt

Outline

- Basic information
- Accelerator Model
- Execution Model
- Memory Model
- Programming Model

Agenda

- **Basic information**
- Accelerator Model
- Execution Model
- Memory Model
- Programming Model

Balance between Productivity & Flexibility



- Accelerated **libraries**:

- Small code changes
- Limited by what libraries are available
- High performance

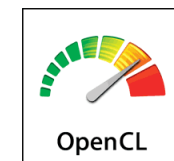


- Compiler **directives** **OpenACC**
Directives for Accelerators
 - Simple and familiar
 - Directives not always expose low level details



- Parallel programming **languages**

- Expose low level details for maximum performance
- Often difficult to learn



OpenACC is an accelerator programming model

- **Pragma based** accelerator programming model
 - **Compiler directives** to specify parallel regions in C, C++, Fortran
 - Programming model allows programmers to **start simple**
 - Similar to the use of **OpenMP**
 - **High level programming** model for accelerator based architectures
 - Create **heterogeneous programs** without explicit accelerator initialization

OpenACC is interoperable and portable

- **Interoperable** with Accelerator Programming Languages and Libraries
- **Portable** across OSes, host CPUs, accelerators and compilers
- **Developed by**

The Portland Group



Where to get information about OpenACC?

- **OpenACC Website:**

- Specification, quick reference card, getting started videos, Hands-on Labs
 - <http://www.openacc.org>
- Best practice guide:
 - <http://www.openacc.org/content/openacc-programming-best-practices-guide>

- **Website from NVIDIA:**

- OpenACC Toolkit
 - <https://developer.nvidia.com/openacc>
- Getting started information
 - <https://developer.nvidia.com/how-to-openacc>

OpenACC has its benefits and limitations

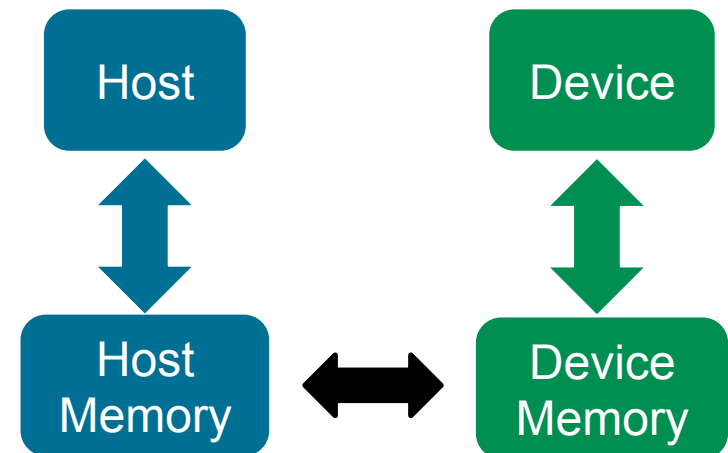
Open	Open standard for GPU programming
Productive	Using only compiler directives to accelerate your application
Portable	Same code for a wide range of architectures
Powerful	Complete access to massive parallel power of accelerators
Performance	Simplicity and portability sometimes may limit the performance

Agenda

- Basic information
- **Accelerator Model**
- Execution Model
- Memory Model
- Programming Model

OpenACC has an abstract model

- To ensure portability OpenACC has an **abstract model** for accelerated computing
- **Offloading** from host to device
- Host and device could be the same
- Single or separate **memory spaces**

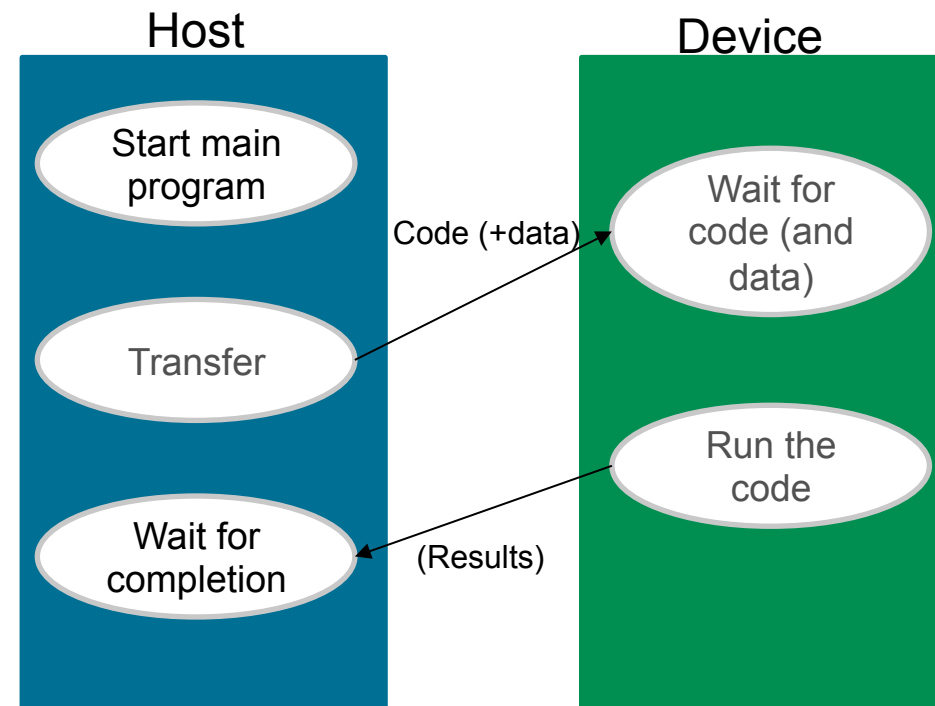


Agenda

- Basic information
- Accelerator Model
- **Execution Model**
- Memory Model
- Programming Model

OpenACC executes host-directed

- **Main** program runs on **host**
- Code is **transferred** to the accelerator
- **Execution** on accelerator is **started**
- **Wait** for completion

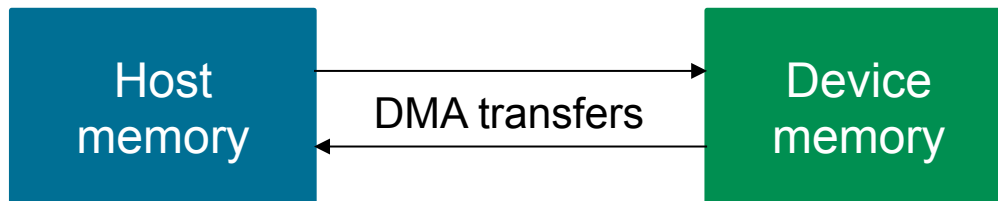


Agenda

- Basic information
- Accelerator Model
- Execution Model
- **Memory Model**
- Programming Model

Memory needs to be transferred

- There may be **two separate memory spaces**



- The **data** needed for the calculations has to be **transferred to the device**
- The **host** has to **read** the results back

Memory transfer is hidden, thus be cautious

- **Transfer hidden** from the programmer, so beware:
 - Latency
 - Bandwidth
 - Limited device memory size

- On the accelerator:
 - Device Memory and Host memory are **not coherent**
 - **Memory management** can be done by compiler

Agenda

- Basic information
- Accelerator Model
- Execution Model
- Memory Model
- **Programming Model**

Programming by directives

- **Directive based**
 - Activated by **compiler options**
 - For example: `$ pgcc -acc`
 - **Ignored** if not supported by compiler
 - Allows productive, incremental and single-source **porting**
- **Features** through advanced directives or library functions
 - Explicit **data and device management**
 - **Asynchronous** kernel execution

Directives are simple and portable

- **Simple** compiler hints specified by using the **#pragma** mechanism (for C/C++) or **!\$** (for Fortran)
- Should be familiar to **OpenMP** users
- Work on **GPUs**, **multicore** CPUs or other **accelerators** like Intel Xeon Phi
- Syntax (**C/C++**):
#pragma acc directive [clause [, clause] ...] new-line
- Syntax (**Fortran**):
!\$acc directive [clause [, clause] ...]
!\$acc end directive

Example

C code

```
#pragma acc data
    copy(x[0:N],y[0:N])
#pragma acc parallel loop
{
    for (int i=0; i<N; i++){
        x[i] = 1.0;
        y[i] = 2.0;
    }
    for (int i=0; i<N; i++){
        y[i] = i*x[i]+y[i];
    }
}
```

Fortran code

```
!$acc data copy(x(1:N),y(1:N))
!$acc parallel loop

    do i=1,N
        x(i) = 1.0
        y(i) = 2.0
    end do

    do i=1,N
        y(i) = i*x(i)+y(i)
    end do

!$acc end parallel loop
!$acc end data
```