OpenMP and OpenACC a Comparison

James Beyer
Cray Inc.

Outline

- **Related talks here at GTC**
- **Background**
  - OpenMP
  - OpenACC
- **Important differences (today)**
  - Parallelism
  - Present_or_*
  - Scalars
  - Loops
  - Unstructured data
  - Calls (separate compilation units)
  - Nested parallelism
- **What is next**
  - OpenMP
  - OpenACC

Related talks here at GTC

- 4438 - What's new in OpenACC 2.0 and OpenMP 4.0
- S4474 - Scaling OpenACC Across Multiple GPUs
- S4472 - Performance Analysis and Optimization of OpenACC Applications
- S4514 - Panel on Compiler Directives for Accelerated Computing
- Hangout: OpenACC

- There are more just search for OpenACC

Background -- OpenMP

- FORTRAN version 1.0 - (October 1997)
- Accelerator additions
  - Proposal submitted Dec 2009
  - Subcommittee formed Aug 2009
- Cray OpenMP for Accelerators nears release
- Fall 2010 several members for OpenACC working group
- TR1 - Technical Report on Directives for Attached Accelerators (November 2012)
- OpenMP 4.0 (July 2013)

Background -- OpenACC

- PGI releases accelerator directives
- CAPS releases HMPP
- Fall 2010 several members form OpenACC working group
- OpenACC 1.0 (Nov 2010)
- OpenACC 2.0 (June 2013)

Important differences

- Parallelism
- Present_or_*
- Scalars
- Loops
- Calls (separate compilation units)

Parallelism

- **OpenACC**
  - "Off-load" and parallel startup tied together
    - Acc parallel
    - Acc kernels
- **OpenMP**
  - "Off-load" and parallel startup disconnected
    - Omp target
      - Omp parallel
      - Omp teams

Parallel startup example (Fortran)

OpenACC

```
!$acc parallel
…
!$acc end parallel

Or

!$acc kernels
…
!$acc end kernels
```

OpenMP

```
!$omp target
!$omp teams/parallel
…
!$omp end teams
!$omp end target
```

Parallel startup example (C/C++)

OpenACC

```
#pragma acc parallel
{
…
}

Or

#pragma acc kernels
{
…
}
```

OpenMP

```
#pragma omp target
#pragma omp teams/parallel
{
…
}
```

OpenMP teams vs parallel

- **Why two different “parallel” mechanisms**
- **Teams**
  - Independent collision domains
  - Same behavior as OpenACC gangs
  - Only select directives allowed
- **Parallel**
  - A single collision domain
  - Default if neither is present
  - All non-accelerator OpenMP directives allowed

Present_or_*

- **OpenACC**
  - present_or_* programmer visible
    - Copy, copyin copyout, create
  - Copy* without present allowed
    - Error prone
    - Hard to debug
    - Little actual savings
- **OpenMP**
  - present-or_* not programmer visable
  - map always implies present test
    - In, out, inout, allocate

Scalars

- **OpenACC**
  - Firstprivate by default
  - User can override
    - Error prone
  - Allows implementation to make these kernels arguments
  - Pointers are “special”
- **OpenMP**
  - No such restrcition
  - Pointers are scalars

Loops

- **OpenACC**
  - One construct “loop”
  - Multiple parallelism types
  - “nested” parallelism implicit
  - Three levels available
    - Gang
    - Worker
    - vector
- **OpenMP**
  - Three constructs
    - Distribute
    - Do/for
    - Simd
  - Nested parallelism explicit

Loop examples

OpenACC

```
!$acc loop
do i=1,n
…
enddo
```

OpenMP

```
!$omp do
or
!$omp distribute
do i=1,n
…
enddo
!$omp end distribute
or
!$omp end do
```

Loop examples

OpenACC

```
!$acc loop
do i=1,n
!$acc loop
  do j=1,m
    …
  enddo
enddo
```

OpenMP

```
!$omp distribute
do i=1,n
!$omp parallel do
  do j = 1,m
    …
  enddo
!$omp end parallel do
enddo
!$omp end distribute
```

Loop examples

OpenACC

```
!$acc loop gang worker vector
do i=1,n
...
enddo
```

OpenMP

```
!$omp distribute parallel do simd
do i=1,n
...
enddo
!$omp end distribute parallel do simd
```

Unstructured data

- **Separate the move to and the move from parts of data constructs**
- **Enter data**
  - Constructors
- **Exit data**
  - destructors
- **OpenACC**
  - Added support in 2.0
- **OpenMP**
  - Nearing completion of feature

Calls

- **OpenACC**
  - Routine
  - Only one type of parallelism allowed
    - Gang
    - Worker
    - Vector
    - Seq
  - Hard on user
  - Easy for implementer
- **OpenMP**
  - Declare
    - Type of parallelism ignored
  - Easy on user
  - Hard for implementer

Nested parallelism

- **OpenACC**
  - Added in 2.0
  - Currently no full implementations
    - Why?

- **OpenMP**
  - Parallel inside of teams is allowed
  - Teams inside of teams is not allowed.

What is next

- **OpenACC**
  - Tools interfaces
  - Better user defined type support
  - …
- **OpenMP**
  - What is next
  - Unstructured data
  - Declare target deferred_map
  - Interoperability with accelerated libraries
  - Multiple devices
  - User defined type support


CRAY
CRAY
CRAY
CRAY
CRAY
CRAY
CRAY XC30


CRAY XE6


CRAY XK7