Introduction to OpenACC Data Management

Stefano Markidis

Four Key-Points

1. OpenACC data regions enable the programmer to leave data in GPU device memory and reuse it across multiple procedures.
2. Data regions can be implicit if the compiler automatically take care of copying in and out memory objects to/from GPU memory
3. Data regions can be explicit when we tell compiler which memory object to move in and out and leave on GPU using `#pragma acc data`
4. We use clause `copyin`, `copyout`, `present` and `copy` to tell compiler which memory object to move to/from GPU memory

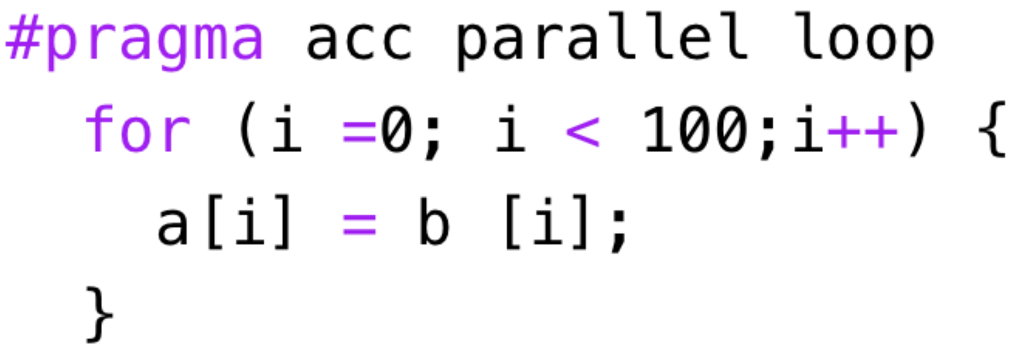
OpenACC Memory Model

- The host memory and the device memory are treated as separated.
    - host is not able to access device memory directly
    - the device is not able to access host memory directly.
- Data needs to be transferred from the host to the device before kernel
    - We can annotate which memory objects need to be transferred.
- The OpenACC compiler will automatically generate code for memory allocation, copying, and de-allocation.

OpenACC Data Management

- We know that the device has a completely separate memory and therefore we need to be careful about where the data we are operating on is.
- OpenACC has two modes of dealing with data:
  - **Implicit Data Regions**: where we let the compiler decide what to do
  - **Explicit Data Regions**: where we tell the compiler what to do

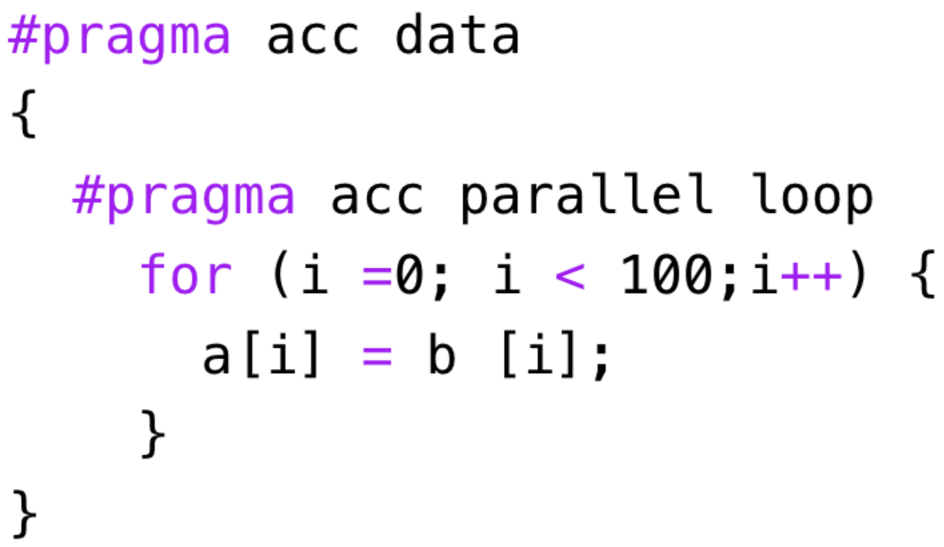
Implicit Data Regions



```
#pragma acc parallel loop
  for (i =0; i < 100;i++) {
    a[i] = b [i];
  }
```

- This is an implicit data region:
  - We didn’t tell the compiler to do anything with the data.
  - If the compiler can determine the size of `a` and `b`, it can copy them to the device along with the kernel.
- Scalars like `i` and constants are copied for us

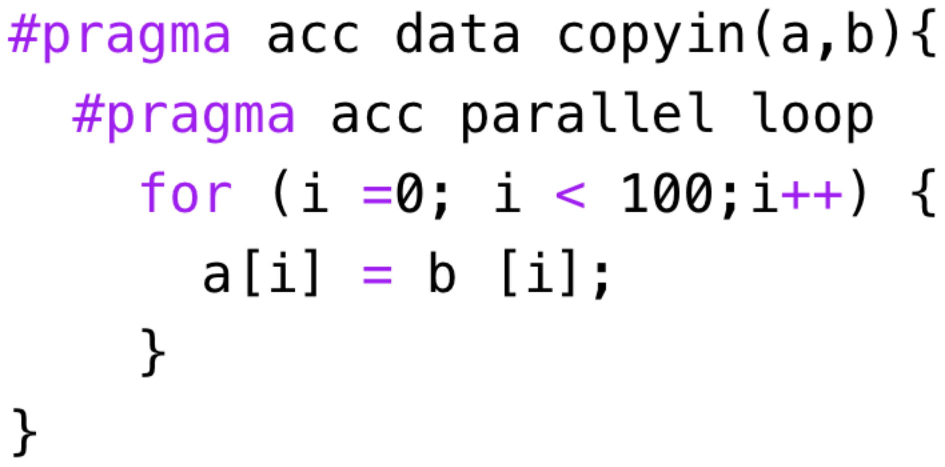
Explicit Data Regions



```
#pragma acc data
{
  #pragma acc parallel loop
    for (i =0; i < 100;i++) {
      a[i] = b [i];
    }
}
```

- We added the `#pragma acc data` line with two braces.
  - This defines a data region
- Data copied on to the device within this region will persist with the region.
  - **Data regions** enable the programmer to **leave data** in GPU device memory **and re-use it** across multiple procedures.

Explicit Data Regions II



```
#pragma acc data copyin(a,b){
  #pragma acc parallel loop
    for (i =0; i < 100;i++) {
      a[i] = b [i];
    }
}
```

- We added the `copyin` clause.
  - The compiler will now copy the arrays `a` and `b` to the device.
    - Compiler still has to work out sizes.
- Could use `a` or `b` later inside the data region for another kernel without having to copy in again

Other Data Clauses

- `copyout`: Create an array on the device and copy to the host at the end of the region
  - we need to initialize the array
- `create`: Create an array on the device (good for temporary arrays).
- `present`: The data is already on the device (used in function calls).
- `copy`: Copy data to and from the device.
  - Use sparingly! You generally only need to copy big data to the device and a small result back

To Summarize

- OpenACC data regions enable the programmer to leave data in GPU device memory and reuse it across multiple procedures.
  - We use clause `copyin`, `copyout`, `present` and `copy` to tell compiler which memory object to move to/from GPU memory
- Data regions can be implicit if the compiler automatically take care of copying in and out memory objects to/from GPU memory
- Data regions can be explicit when we tell compiler which memory object to move in and out and leave on GPU using `#pragma acc data`