Nvidia GPU Technology Conference
OpenACC Data Management - Hands-On Session (S3532)

Rengan Xu
uhxrg@cs.uh.edu

University of Houston
Dept. of Computer Science

21 Mar 2013

Getting Started

- Log into your Nvidia account

Getting Started

- Log into your Nvidia account
- Use the PGI 12.10 module

  ```
  module load pgi/12.10
  ```

Getting Started

- Log into your Nvidia account
- Use the PGI 12.10 module
  `module load pgi/12.10`
- Find the lab code in
  `/sp_c_lab`

Getting Started

- Log into your Nvidia account
- Use the PGI 12.10 module
  `module load pgi/12.10`
- Find the lab code in
  `/sp_c_lab`
- Make the serial version
  `make CLASS=A STEP=ser`

NAS-SP benchmark

- SP is derived from "Scalar Pentadiagonal", the code solves a system of nonlinear PDEs using three different algorithms

NAS-SP benchmark

- SP is derived from "Scalar Pentadiagonal", the code solves a system of nonlinear PDEs using three different algorithms
- The SP benchmark is a serial C version of the NPB SP code, developed by the Center for Manycore Programming at Seoul National University and derived from the serial Fortran versions in "NPB3.3-SER" developed by NAS

NAS-SP benchmark

- SP is derived from "Scalar Pentadiagonal", the code solves a system of nonlinear PDEs using three different algorithms
- The SP benchmark is a serial C version of the NPB SP code, developed by the Center for Manycore Programming at Seoul National University and derived from the serial Fortran versions in "NPB3.3-SER" developed by NAS
- SP's workflow:
  extract_rhs()$\rightarrow$ *initialize*() $\rightarrow$ *adi*() $\rightarrow$ *initialize*() $\rightarrow$ *adi*() $\rightarrow$ *verify*()

In adi(): compute_rhs()$\rightarrow$ *txinvr*() $\rightarrow$ *x_solve*() $\rightarrow$ *y_solve*() $\rightarrow$ *z_solve*() $\rightarrow$ *add*()

UNIVERSITY of HOUSTON

Scope of the lab: OpenACC Data Management

- Add local data regions

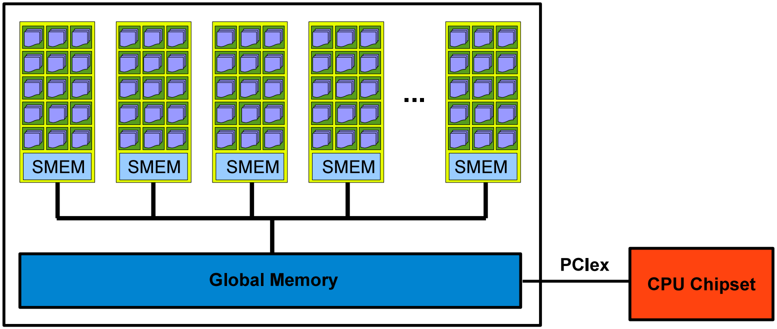
Scope of the lab: OpenACC Data Management

- Add local data regions
- Add data regions across routines

Scope of the lab: OpenACC Data Management

- Add local data regions
- Add data regions across routines
- Update host/device data from device/host

GPU Architecture



OpenACC Data Movement

- Start by checking the performance of base OpenACC lab package

```
cp -r src_acc2 src_myacc2
make CLASS=W STEP=myacc2
```

OpenACC Data Movement

- Start by checking the performance of base OpenACC lab package

  ```
  cp -r src_acc2 src_myacc2
  make CLASS=W STEP=myacc2
  ```

- Why does the execution take so long?

Compiler Feedback

```
x_solve:
     58, Generating present_or_copy(rhs[1:nz2][1:ny2][0:gp0][0:])
         Generating present_or_copyin(speed[1:nz2][1:ny2][0:nx2+2])
         Generating present_or_copyin(us[1:nz2][1:ny2][0:nx2+2])
         Generating allocate(rhonX[1:nz2][1:ny2][0:])
         Generating copyin(rhonX[1:nz2][1:ny2][0:nx2+2])
         Generating copyout(rhonX[1:nz2][1:ny2][0:gp0])
         Generating present_or_copyin(rho_i[1:nz2][1:ny2][0:gp0])
         Generating copyin(lhsX[1:nz2][1:ny2][0:][0:])
         Generating copyout(lhsX[1:nz2][1:ny2][0:nx2+2][0:])
         Generating copyin(lhsmX[1:nz2][1:ny2][0:][0:])
         Generating copyout(lhsmX[1:nz2][1:ny2][0:nx2+2][0:])
         Generating copyin(lhspX[1:nz2][1:ny2][0:][0:])
         Generating copyout(lhspX[1:nz2][1:ny2][0:nx2+2][0:])
```

OpenACC Data Region Syntax

```
#pragma acc data <clause>
copyin, copyout, copy, create, etc
```

If compiler cannot determine array size, provide array size in data clauses:

```
#pragma acc data copy(a[start:length])
```

Add Data Region to x_solve.c

```
 #pragma acc data pcopy(rhs) pcopyin(us,speed,rho_i)
                  create(lhsX,lhsmX,lhspX,rhonX)
{
     // lots of code
      ninvr(); //our kernel
} // end acc data region
```

Add kernels data clause to ninvr.c

```
 #pragma acc kernels pcopy(rhs)
{
      for (k = 1; k <= nz2; k++) {
            ...
      }
}
```

Assignment - Add data regions/clauses in other files

- Add data regions to files -
  - `y_solve.c`
  - `z_solve.c`
  - `rhs.c`
- Add data clauses to the kernels pragmas in -
  - `add.c`
  - `pinvr.c`
  - `txinvr.c`
  - `tzetar.c`

Note: Test and check for correct answers every time

Present clause

- The `present` clause indicates that the data is already copied to the device
- If it's not there, the program will exit
- Alternatively, we can use `pcopy` clause, `present_or_copy` - If the data isn't there it will copy it
- Adding higher level data region changes the meaning of `pcopy` from `copy` to `present`

Present clause

- `present` clause allows to use data across routines!
- Alert! Your data is NOT automatically synchronized and may result in wrong answers!

Update host/device data

To synchronize host and device arrays, use `update` directive within data region

```
#pragma acc update host(arr)
#pragma acc update device(arr)
```

Add data region to sp.c

```
 #pragma acc data create(rhs)
{
      exact_rhs();
      initialize();
      ...
      adi();
} // end acc data region
timer_stop(1);
```

Assignment

- Add these variables to the data region in `sp.c` - create clause `us, vs, ws, qs, rho_i, speed, square, forcing, and u`

Assignment

- Add these variables to the data region in `sp.c` - create clause `us, vs, ws, qs, rho_i, speed, square, forcing, and u`
- Suggestion: Add the variables one by one to the create clause

Assignment

- Add these variables to the data region in `sp.c` - create clause `us, vs, ws, qs, rho_i, speed, square, forcing, and u`
- Suggestion: Add the variables one by one to the create clause
- Two of these variables will give you wrong answers! Try to figure out why and how to fix them

Assignment

- Add these variables to the data region in `sp.c` - create clause `us, vs, ws, qs, rho_i, speed, square, forcing, and u`
- Suggestion: Add the variables one by one to the create clause
- Two of these variables will give you wrong answers! Try to figure out why and how to fix them
- **Hint:** Fix the data synchronization between the host and device

Optimizing Data Movement

- Check final timings

```
make CLASS=W STEP=mysrc2
```

Optimizing Data Movement

- Check final timings
  `make CLASS=W STEP=mysrc2`
- You should see significant improvement!

Optimizing Data Movement

- Check final timings
  `make CLASS=W STEP=mysrc2`
- You should see significant improvement!
- Compare with lab solutions:
  `make CLASS=W STEP=src2a`

Optimizing Data Movement

- Check <u>final</u> timings
  `make CLASS=W STEP=mysrc2`
- You should see significant improvement!
- Compare with lab solutions:
  `make CLASS=W STEP=src2a`

Code can be found at:
http://www.pgroup.com/lit/samples/labs/gtc_openacc_pgi_labs.zip