PDC Summer School 2015

GPU Programming: OpenACC

2015-08-21

Michael Schliephake

KTH – CSC - HPCViz

OpenACC Overview

- OpenACC
  - Programming model for today's GPUs
  - Fortran, C/C++
  - OpenACC 2.0a since August 2013
  - Information at http://www.openacc-standard.org/
- OpenMP
  - Programming model for ”shared” memory systems for Fortran, C/C++
  - Accelerator support
  - OpenMP 4.0 since July 2013
  - Information at http://openmp.org/

OpenACC Overview

- Execution model
  - Main program executed on host
  - Host controls offload and execution on device
  - Device executes kernels, i.e. loops that are executed as kernels
  - Tasks for host:
    - memory allocation, data and code transfer
    - queuing execution, wait for completion
    - passing arguments and results
  - Sequences of operations typical

OpenACC Overview

- Different memory spaces for host and device
- Weak memory model without synchronisation between SMs (need to use memory barriers) and risk of race conditions in kernels
- OpenACC specifies data movement implicit, i.e. directives instruct compiler and hint cache usage

OpenACC example

```
void matvecmul( float* x, float* a, float* v, int m, int n ) {


    for(int i = 0; i < m; ++i) {
        float xx = 0.0;

        for (int j = 0; j < n; ++j)
            xx += a[i*n+j]*v[j];
        x[i] = xx;
    }
}



{
    …

    {
        …

        matvecmul( x, a, v, m, n );
    }    …
}
```

OpenACC example

```
void matvecmul( float* x, float* a, float* v, int m, int n ) {

#pragma acc parallel loop gang
    for(int i = 0; i < m; ++i) {
        float xx = 0.0;
        #pragma acc loop worker reduction(+:xx)
        for (int j = 0; j < n; ++j)
            xx += a[i*n+j]*v[j];
        x[i] = xx;
    }
}



{
    …

    {
        …
        matvecmul( x, a, v, m, n );
    }    …
}
```

OpenACC example

```
void matvecmul( float* x, float* a, float* v, int m, int n ) {
#pragma acc parallel loop gang \
  copyin(a[0:n*m],v[0:n]) copyout(x[0:m])
    for(int i = 0; i < m; ++i) {
        float xx = 0.0;
        #pragma acc loop worker reduction(+:xx)
        for (int j = 0; j < n; ++j)
            xx += a[i*n+j]*v[j];
        x[i] = xx;
    }
}



{
    …

    {
        …
        matvecmul( x, a, v, m, n );
    }    …
}
```

OpenACC example

```
void matvecmul( float* x, float* a, float* v, int m, int n ) {
#pragma acc parallel loop gang \
  pcopyin(a[0:n*m],v[0:n]) pcopyout(x[0:m])
    for(int i = 0; i < m; ++i) {
        float xx = 0.0;
        #pragma acc loop worker reduction(+:xx)
        for (int j = 0; j < n; ++j)
            xx += a[i*n+j]*v[j];
        x[i] = xx;
    }
}

#pragma acc data copyin(a[0:n*m])
{
    …
    #pragma acc data copyin(v[0:n]) copyout(x[0:n])
    {
        …
        matvecmul( x, a, v, m, n );
    }    …
}
```

OpenACC Calculation of Pi

```
#include <stdio.h>
#define N 1000000

int main( int argc, char *argv[] )
{
    double pi = 0.0f;
    long i;


#pragma acc parallel loop reduction(+:pi)
    for (i=0; i<N; i++) {
        double t= (double)((i+0.5)/N);
        pi +=4.0/(1.0+t*t);
    }


    printf("pi=%16.15f\n",pi/N);
    return 0;
}
```

Compiling for OpenACC

```
$ module load compiler/pgi/13.10

$ pgcc -acc -Minfo=acscel pi .c -o pi
$ ./pi

$ export PGI_ACC_TIME =1
$ ./pi
```

Getting started on your own computer:

https://developer.nvidia.com/openacc