GPU programiranje

Kako iskoristiti GPU bez CUDA-e?

OpenACC

- Standard za paralelno programiranje
- nVidia
- PGI
  - http://www.pgroup.com/
- CRAY
  - http://www.cray.com
- CAPS
  - http://www.caps-entreprise.com

OpenACC

- 'pre-compiler' i 'compiler' instrukcije
- Ubrzanje izvođenja bez dodatnog programiranja
- C ili Fortran
- Prebacuje izvođenje dijela koda na GPU za postizanje boljih performansi
- Nije besplatno :(

OpenACC

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

int main(void) {
    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;
}
```

- Računanje broja PI
- Kod se ne mijenja, samo se dodaje direktiva (crveno)

OpenACC

- Operacija izglađivanja (*smoothing*) 2D matrice
  - 9 točaka (3x3 okolina)
  - 3 parametra
    - Središnji element
    - Susjedni elementi (lijevo, desno, gore, dolje)
    - Dijagonalni elementi

OpenACC

```
void smooth (MAT a, MAT b, float p0, float p1, float p2, int n, int m, int niters)
{
    int i, j, iter;
    for (iter = 1; iter <= niters; iter++) {
        for (i = 1; i < n-1; i++) {
            for (j = 1; j < m-1; j++) {
                a[i][j] = p0*b[i][j]+
                    p1*(b[i-1][j]+b[i+1][j]+b[i][j-1]+b[i][j+1])+
                    p2*(b[i-1][j-1]+b[i-1][j+1]+b[i+1][j-1]+b[i+1][j+1]);
            }
        }
        for (i = 1; i < n-1; i++) {
            for (j = 1; j < m-1; j++) {
                b[i][j] = a[i][j];
            }
        }
    }
}
```

OpenACC

```
void smooth (MAT a, MAT b, float p0, float p1, float p2, int n, int m, int niters)
{
    int i, j, iter;
    for (iter = 1, iter <= niters, iter++) {
        #pragma acc region
        {
        for (i = 1; i < n-1; i++) {
            for (j = 1; j < m-1; j++) {
                a[i][j] = p0*b[i][j]+
                    p1*(b[i-1][j]+b[i+1][j]+b[i][j-1]+b[i][j+1])+
                    p2*(b[i-1][j-1]+b[i-1][j+1]+b[i+1][j-1]+b[i+1][j+1]);
            }
        }
        for (i = 1; i < n-1; i++) {
            for (j = 1; j < m-1; j++) {
                b[i][j] = a[i][j];
            }
        }
        }
    }
}
```

OpenACC

```
void smooth (MAT a, MAT b, float p0, float p1, float p2, int n, int m, int niters)
{
    int i, j, iter;
    #pragma acc data region copy(b[0:n-1][0:m-1]) local(a[0:n-1][0:m-1])
    {
    for (iter = 1; iter <= niters; iter++) {
        #pragma acc region
        {
        for (i = 1; i < n-1; i++) {
            for (j = 1; j < m-1; j++) {
                a[i][j] = p0*b[i][j]+
                    p1*(b[i-1][j]+b[i+1][j]+b[i][j-1]+b[i][j+1])+
                    p2*(b[i-1][j-1]+b[i-1][j+1]+b[i+1][j-1]+b[i+1][j+1]);
            }
        }
        for (i = 1; i < n-1; i++) {
            for (j = 1; j < m-1; j++) {
                b[i][j] = a[i][j];
            }
        }
        }
    }
}
```

nVidia NPP

- NVIDIA Performance Primitives
- Dolazi u kompletu s toolkitom
- Kolekcija funkcija za obradu signala, slike i videa ubrzane korištenjem GPU-a
- 5-10x brze od CPU alternativa

nVidia NPP

- Aritmetičke i logičke operacije
  - Add, Sub, Mul, Div, AbsDiff, Threshold, Compare
- Konverzije formata
  - RGBToYCbCr, YcbCrToRGB, YCbCrToYCbCr, ColorTwist, LUT_Linear
- Filter Funkcije
  - FilterBox, Filter, FilterRow, FilterColumn, FilterMax, FilterMin, Dilate, Erode, SumWindowColumn, SumWindowRow
- JPEG
  - DCTQuantInv, DCTQuantFwd, QuantizationTableJPEG
- Geometrijske transformacije
  - Mirror, WarpAffine, WarpAffineBack, WarpAffineQuad, WarpPerspective, WarpPerspectiveBack , WarpPerspectiveQuad, Resize
- Statistika
  - Mean_StdDev, NormDiff, Sum, MinMax, HistogramEven, RectStdDev

nVidia NPP

- Primjer – Box Filtering
- Filter
  - NxM matrica
  - Moguće definirati razne operacije
    - Izglađivanje (*smoothing*)
    - Izoštravanje (*sharpen*)
    - Detekcija rubova (*edge detection*)

cuFFT

- NVIDIA CUDA Fast Fourier Transform library
- Algoritmi bazirani na Cooley-Tukey i Bluestein algoritmima
- Performanse GPU-a bez potrebe za razvijanjem vlastite GPU FFT implementacije
- Dolazi u kompletu s toolkitom

cuFFT

- 1D, 2D, 3D transformacije za kompleksne i realne tipove podataka
- 1D transformacije na vektorima veličine do 128 milijuna elemenata
- API sličan FFTW Advanced Interface
- Asinkrono izvođenje
- *Single* i *double precision floating point* formati
- "*Thread-safe*" te je moguće pozivati funkcije iz više konkurentnih dretvi

cuBLAS

- NVIDIA CUDA Basic Linear Algebra Subroutines
- 6x do 17x brže od MKL BLAS
- Dolazi u kompletu s toolkitom

cuBLAS

- Potpuna podrška za sve 152 standardne BLAS funkcije
- Single i double precision floating point za realne i komplekse tipove podataka
- Podržava i Fortran
- Podrška za multi GPU i konkurentne CUDA *kernele*
- *Batched* GEMM API

CRO-NGI

- Pristupni čvor ui-64.cro-ngi.hr
- Podnošenje poslova
  - Certifikat korisnika CRO-NGI
    - www.cro-ngi.hr
  - Condor G

CRO-NGI

- Podnošenje poslova
  - `condor_submit <JDL_skripta>`
  - vraća ID posla
- Zaustavljanje poslova
  - `condor_rm <JOB_id>`
- Stanje poslova
  - `condor_q`
  - `condor_q <JOB_id>`
  - `condor_q -l <JOB_id>`

CRO-NGI

```
executable = deviceQuery.sh
output = deviceQuery.out.$(Process)
error = deviceQuery.err.$(Process)
grid_resource = gt2 ce2.srce.cro-ngi.hr/jobmanager-sge
environment = SGE_PE=gpu
should_transfer_files = YES
transfer_executable = YES
transfer_input_files=deviceQuery.sh, deviceQuery
WhenToTransferOutput = ON_EXIT
```

CRO-NGI

```
#!/bin/sh
FREEGPU="`cat $TMPDIR/gpumachine`"
export CUDA_VISIBLE_DEVICES=$FREEGPU

chmod +x deviceQuery

./deviceQuery
```