HP-SEE

Паралелно програмиране с CUDA

www.hp-see.eu

Емануил Атанасов
Секция “Грид технологии и приложения”
Институт по информационни и комуникационни технологии
Българска академия на науките
emanouil@parallel.bas.bg

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

Outline

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Какво е GPGPU
- Кога можем да използваме GPGPU
- Стартиране на изчислителни ядра при CUDA
- Модел на паметта
- Подходи за паралелизация
- Версии на CUDA
- Програмни езици и поддръжка
- Други възможности за използване на GPGPU и CUDA
- Заключения

Какво е GPGPU

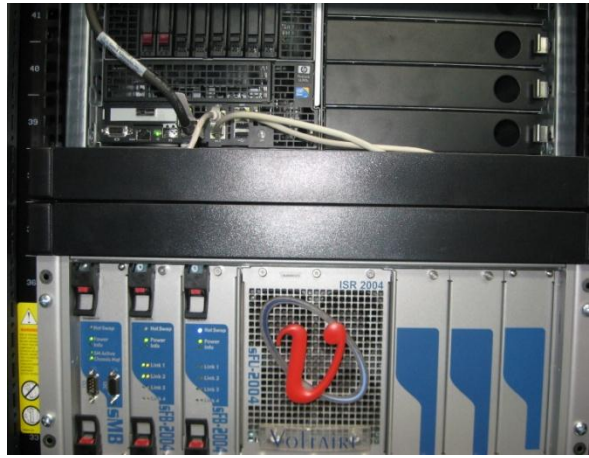
HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- GPGPU означава използване на графични карти (GPU) за изчисления от общ характер.
- При това се използва големия брой транзистори и съответно изчислителни елементи, налични в графичните карти, за извършване на изчисления, които обикновено се правят на централния процесор
- Съвременните графични карти имат по-голям брой транзистори и много по-високо ниво на паралелизъм на от CPU, което води до по-добра ефективност от гледна точка на цена, енергопотребление, заемано място.
- За GPGPU изчисления могат да се използват масови карти, използвани предимно за компютърни игри, но основните производители NVIDIA и AMD (ATI) предоставят специализиран хардуер за инсталацията на HPC (Tesla, Fermi, Kepler, AMD FireStream)
- Ключови технологии за разработката на софтуер
- OpenCL – стандарт за паралелно програмиране, който може да се използва също за паралелна обработка върху много-ядрени процесори
- OpenACC – нов стандарт за паралелно програмиране с използване на директиви, подобно на OpenMP

Кога можем да използваме GPGPU

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- За да използваме GPGPU трябва да разполагаме с хардуер и софтуер
- Няколко начина за използване на предимствата на GPGPU
  - Използване на цяло приложение – GROMACS, NAMD
  - Използване на библиотеки – FFT, BLAS
  - Пренаписване на собствен код
- Може да се използват различни нива на паралелизъм
  - Една карта
  - Няколко карти на един сървър
  - Няколко сървъра с по няколко карти
- При това CUDA може да се комбинира с някой от стандартите за паралелно програмиране MPI или OpenMP



Кога можем да използваме GPGPU

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

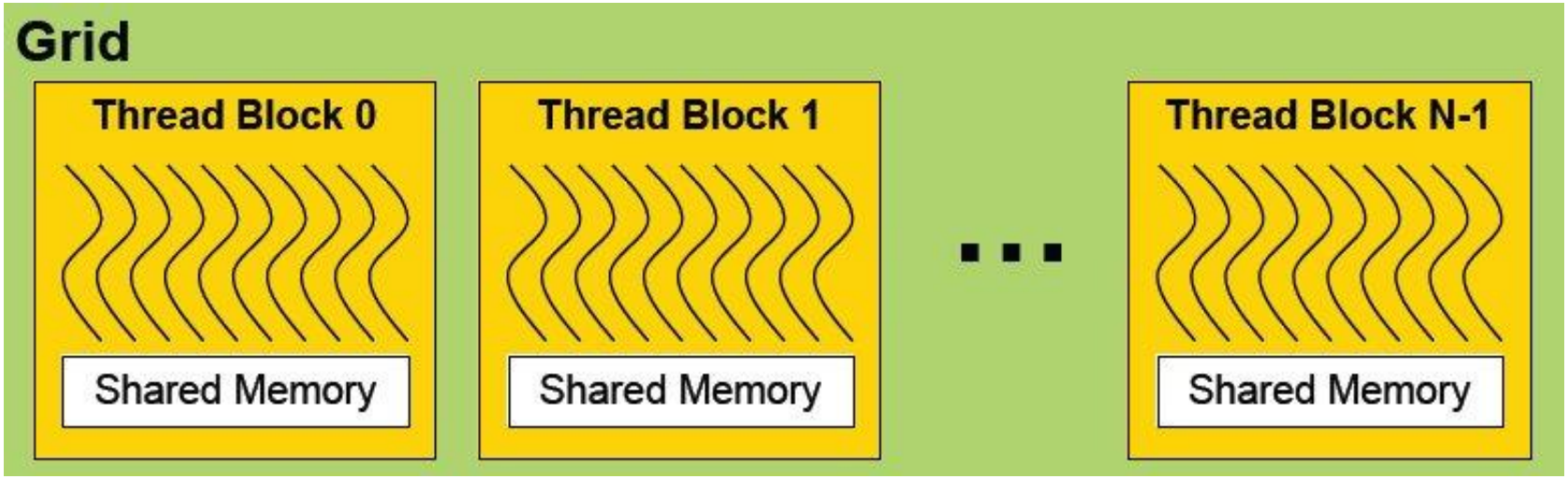
- Области, където се използва GPGPU
  - Финансова математика
  - ДНК анализ
  - Търсене на нефт и газ
  - Сеизмология
  - Други - http://www.nvidia.com/object/cuda_app_tesla.html

- Компютри в Топ 500, използващи NVIDIA – около 50, включително номер 1 – Cray XK7 Titan

Стартиране на изчислителни ядра

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- В терминологията на CUDA изчислително ядро (kernel) е функция, която се изпълнява паралелно върху GPU
- Изчисленията се разпределят между блокове от нишки.
- Един и същи код се изпълнява върху всяка нишка, като нишките имат идентификатори - threadID



Стартиране на изчислителни ядра

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Функциите, които се изпълняват върху GPU, се маркират с ключовите думи __device__ или __global__
- Функциите, които се изпълняват върху CPU, могат да се маркират с ключовата дума __host__
- От __host__ функция може да се извиква само фунция от тип __global__, като при извикването се заявява геометрията на разпределението на нишките по блокове.
- От функция тип __global__ или __device__ може да се извиква фунция тип __device__, но не и фунция тип __host__

Стартиране на изчислителни ядра

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- При стартиране на функция __global__ се указва броят блокове и броят нишки в един блок
- gpukernel<<<dim3 dG, dim3 dB>>>(…)
- dG – размерност на един Grid, състоящ се от блокове
  - двумерен при старите карти, тримерен при новите – dG.x*dG.y*dG.z
- dB – размерност на един блок, състоящ се от нишки
  - тримерен – dB.x * dB.y * dB.z
- Размери, които не са заявени, по подразбиране са 1.
- Размерността на dB трябва да съответства максимално на задачата, а размерността на dG – да бъде достатъчно голяма

- **dim3 grid(16,80), block(16,8);**
- **gpukernel<<<grid,block>>>(…);**
- **gpukernel<<<32,512>>>(…);**
- Параметрите се предават от CPU към GPU, но не трябва да превишават определен общ размер

Стартиране на изчислителни ядра

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Пример за код на ядро и неговото извикване:

```
__global__ void multip(float A[N][N], float B[N][N],
float C[N][N]){
int i = threadIdx.x;
int j = threadIdx.y;
C[i][j] = A[i][j] * B[i][j];
}
int main() {
// 1 block of N * N * 1 threads
int blocks = 1;
dim3 threadsinblock(N, N);
multip<<<numBlocks, threadsinblock>>>(A, B, C);
}
```

Остава да изясним какво означават A,B,C

Модел на паметта

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Паметта на централния процесор е различна от паметта на графичния процесор.
- Паметта на GPU е по-скъпа, по-производителна, но обикновено по-малка по размер
- Сървърните GPU поддържат ECC RAM
- Паметта на GPU е с висока латентност, но с голяма широчина (bandwidth)
- Кешът не работи по същия начин, както при CPU
- От гледна точка на CUDA основното количество памет се определя като глобална памет (__global__), която е споделена между всички нишки
- Сравнително малко количество памет се определя като локална (__shared__), споделена между нишките от един блок
- Всяка нишка има собствени регистри
- Регистрите и паметта от тип __shared__ имат много малка латентност на достъпа, за разлика от паметта тип __global__
- Общото количество на памет от тип __shared__ и регистрите в един блок е много ограничено.
- Указателите към глобална GPU памет са несъвместими с тези за __host__

Модел на паметта

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- При адресирането на паметта кодът обикновено използва идентификаторите, които отличават отделните нишки
- Всички __global__ и __device__ функции имат достъп до следните променливи
  - dim3 gridDim; - размерност на един Grid в блокове (най-много 2D)
  - dim3 blockDim; - размерност на блок от нишки (3D)
  - dim3 blockIdx; - индекс на блока в един grid (има .x и .y)
  - dim3 threadIdx; - индекс на нишка в блока (има .x, .y и .z)

Подходи за паралелизация

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Ако изчисленията в един цикъл са сравнително независими:
  - Копиране на данните от CPU към глобалната GPU памет
  - Извикване на __global__ функция
  - Копиране на резултатите от глобалната GPU памет към CPU
- Трябва да се търси максимизация на броя на блоковете и нишките, като се има предвид ограничението за броя на регистрите и __shared__.
- Когато съседни нишки обменят данни, такива данни най-добре да се заредят в __shared__ памет.

Подходи за паралелизация

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

```
#include <cuda.h>
__global__ void sum_test(int N,double *full_result, double*result){
        unsigned int tid = blockIdx.x * blockDim.x + threadIdx.x;
        unsigned int bid = blockIdx.x;
        double cf=exp((double)tid/(double)N);
        full_result[tid]=cf;
        __shared__ double  s_data[64];
        s_data[threadIdx.x]=cf;
        for ( int dist = blockDim.x/2; dist > 0; dist /= 2 ) {
          if ( threadIdx.x < dist ){
            s_data[ threadIdx.x ] += s_data[ threadIdx.x + dist ];
          }
            __syncthreads( );
        }
        if (threadIdx.x==0){
            result[bid]=s_data[0];
        }
}
```

Подходи за паралелизация

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

```
int main(int argc,char**argv){
        int N=1280;        int gridsize=20;        int numthreads=64;
        dim3 grid=dim3(gridsize,1,1);        dim3 block=dim3(numthreads,1);
        double * full_results_h=(double*)malloc(sizeof(double)*N);
        double * full_results_d, *results_d;
        cudaMalloc(&full_results_d,sizeof(double)*N);
        double * results_h=(double*)malloc(sizeof(double)*gridsize);
        cudaMalloc(&results_d,sizeof(double)*gridsize);
        sum_test<<<grid,block>>>(N,full_results_d, results_d);
        cudaMemcpy(full_results_h, full_results_d,sizeof(double)*N, cudaMemcpyDeviceToHost);
        cudaMemcpy(results_h, results_d,sizeof(double)*gridsize, cudaMemcpyDeviceToHost);
        double full_s, s;
        int i;
        for (i=0,s=0.;i<gridsize;i++)  s+=results_h[i];
        for (i=0;i<N;i++)  full_s+=full_results_h[i];
        printf("%g %g \n",s/N,full_s/N);
        return 0;
}
```

Управление на GPU

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Примери за управление на GPU

```
int deviceCount;
cuDeviceGetCount(&deviceCount);
int device;
for (int device = 0; device < deviceCount; ++device){
  CUdevice cuDevice;
  cuDeviceGet(&cuDevice, device);
  int major, minor;
  cuDeviceComputeCapability(&major, &minor, cuDevice);
}
```

Compute Capability – означава възможност на GPU да поддържа определен тип изчисления

- Например при 1.3 и нагоре може да се смята с **double**
- Чрез опции на компилатора може да се укаже да създаде код, използваш такива възможности

Възможности в CUDA 4.0

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Споделяне на едно GPU между множество CPU нишки
- Една нишка може да достига всички GPU
- No-copy pinning of host RAM
- NVIDIA GPU Direct 1.0:
  - Директен достъп до GPU памет от други устройства (Infiniband cards)
- NVIDIA GPU Direct 2.0:
  - Peer-to-peer access
  - Peer-to-peer transfers

Възможности в CUDA 5.0

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Динамичен паралелизъм – възможност за стартиране на нови паралелни задачи от __global__ и __device__ функции
- Подобрения в компилацията и свързването
- GPUDirect поддържа RDMA

Библиотеки и софтуер за CUDA

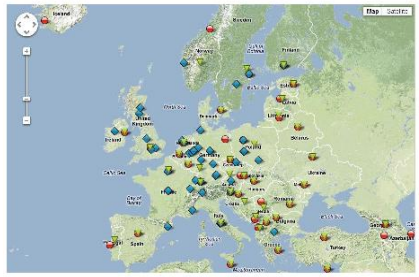
HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- FFT: libcufft
- BLAS: libcublas
- Random numbers: libcurand
- Sparse matrix manipulations: libcusparse

- Debugger – cuda-dbg
- NVIDIA Nsight – Visual Studio и Eclipse
- Приложен софтуер: NAMD, ABINIT, parts of WRF,GROMACS
- Поддръжка на други програмни езици освен C++:
  - Python – pyCUDA
  - Matlab

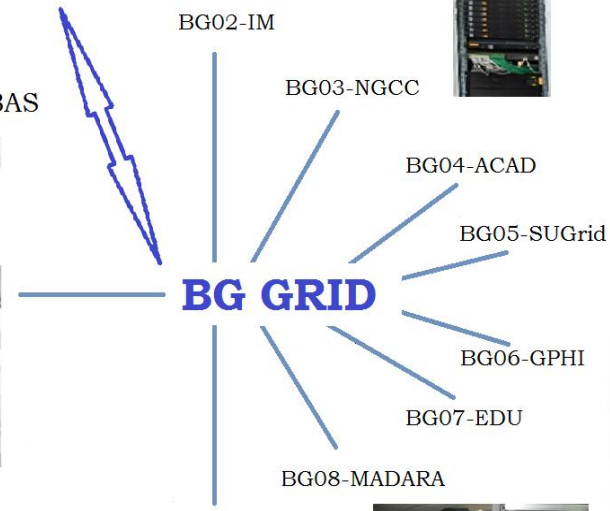
Изчислителни ресурси

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities



Supercomputer BlueGene/P

Grid Cluster at IICT-BAS



ИИКТ-БАН координира работата на Българския грид консорциум, който включва институти от БАН и университети.

Българските грид клъстери са включени в Европейската грид инициатива

Изчислителни ресурси

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

Технически характеристики на високопроизводителния клъстер в ИИКТ-БАН:

- **3 HP Cluster Platform Express 7000 шасита, 36 блейда BL 280c, dual Intel Xeon X5560 @ 2.8Ghz (total 576 cores), 24 GB RAM**
- **8 HP DL 380 G6 сървъра, dual Intel X5560 @ 2.8 Ghz, 32 GB RAM**
- **Voltaire Grid director 2004 non-blocking DDR Infiniband комутатор**
- **Два SAN комутатора, осигуряващи достъп до общо 96 TB дискова памет в два дискови масива**
- **Малък клъстер wn017-wn020 с общо 4 карти NVIDIA GTX 295**
- **Два сървъра 2 HP ProLiant SL390s G7, dual Intel(R) Xeon(R) CPU E5649 @ 2.53GHz, 7 NVIDIA Testla M2090**

Изчислителни ресурси

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- 1 карта M2090 има 512 графични ядра, което осигурява 1331 Gigaflops single precision, 665 Gigaflops double precision, 6 GB ECC GDDR5 RAM, 177 GBytes/sec bandwidth
- В момента имаме 7 карти, т.е. над 9 Teraflops single precision.
- Сървърите са свързани с главния клъстер със същата неблокираща инфинибанд връзка
- Софтуер за развитие на приложения
  - Компилатори от Интел (лицензиран) и NVIDIA (безплатен)
  - PGI компилатор с поддръжка на MPI и OpenACC (лицензиран).

Заключения

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- С помощта на GPGPU определен клас проблеми може да се атакува с изчислителна мощ, сравнима със суперкомпютрите от близкото минало
- Могат да се използват различни подходи за получаване на предимствата на GPGPU, в зависимост от възможностите на екипа и спецификата на предметната област.