はじめてのCUDA
～CUDA事始め～

はるにゃん
Lv1くまー

CUDAとは

- GPGPU計算を行うため
  NVIDIAによるC言語開発環境
- GPGPUとは
  →General Purpose computing on GPU
  →GPUによる汎用目的計算

普通の計算

なんでGPUを使うの？

CPUのコア数は？

→1〜

GPUのコア数は？

→8〜

→SLI(Scalable Link Interface)を使えば、
さらにコアの数は2倍3倍に！！

GPUコア数っていくら？

GeForce	コア数
8600GT	32
8800GT	112
9600GT	64
9800GT	112
9800GTX/GTX+	128
9800GX2	256
GTX 280	240
GTX 295	480
9400M	16

すごい多い！

NVIDIAのCUDAに対する説明は？

- GPUで並列アプリケーション開発を行うための標準C 言語
- FFT（高速フーリエ変換）およびBLAS（線形代数の基本サブルーチン）用標準数値ライブラリ
- GPU とCPU間での高速データ転送パスを使用したコンピューティング専用CUDAドライバ
- OpenGLおよびDirectXグラフィックスドライバとCUDAドライバを同時使用可能
- Linux 32/64ビット、Windows XP 32/64ビット、およびMacのOSをサポート

ちょwおまwVistaは？www

AMD(ATI)の動向は？

- ATI Stream
- ATIが動画変換ソフトを提供
- でも？あんまり聞かない
- GPGPUで検索しても？出てこない

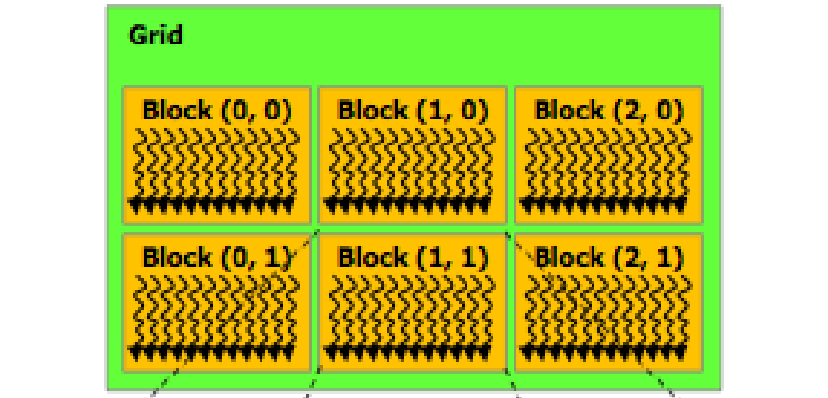
どーなってんのー？w

CUDAとは

- NVIDIAのGPGPU開発環境
- GPGPUを使えば、
  超マルチスレッドプログラミングが可能
- 一般のご家庭に
  スーパーコンピューターをご提供しますw

用語の定義

- Grid = Block × N
- Block = Thread × M
- Multiprocessor = Core / 8



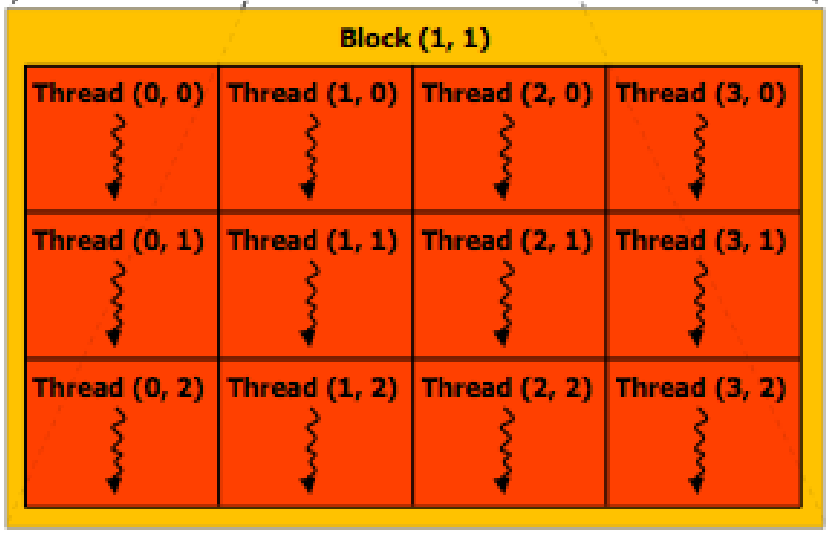
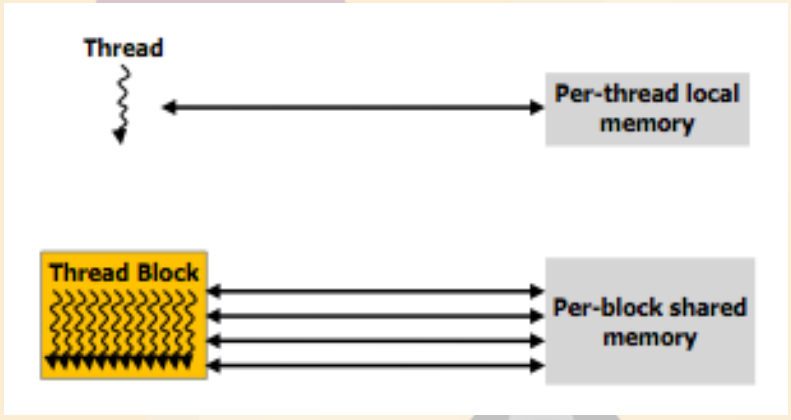


Figure 2-1. Grid of Thread Blocks



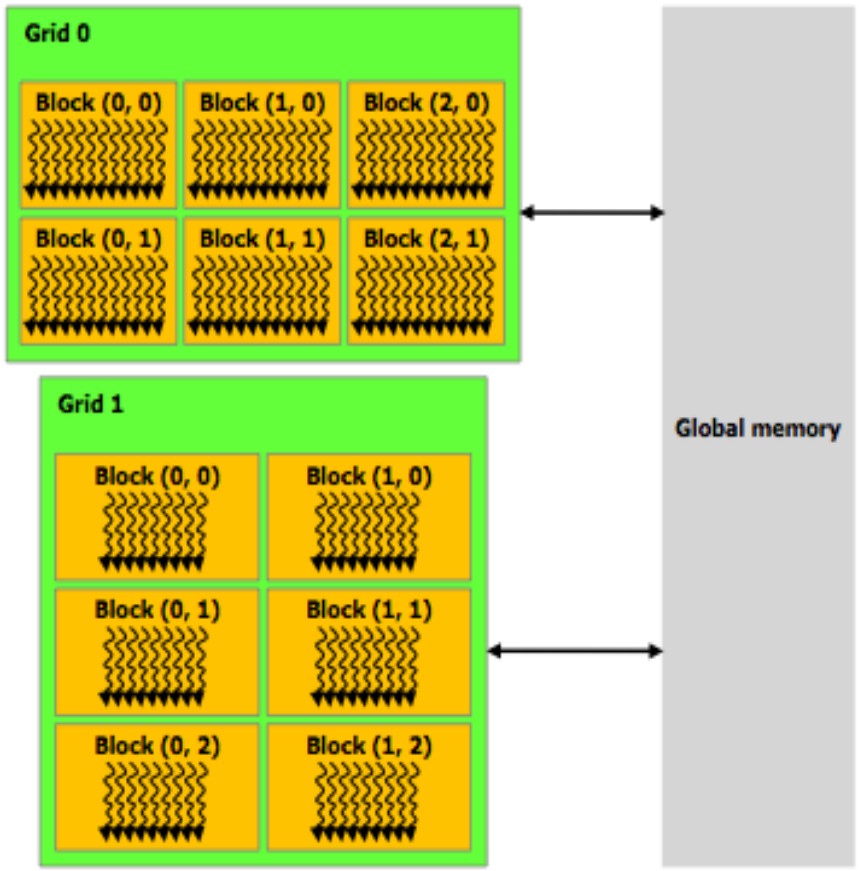


Figure 2-2. Memory Hierarchy

並列足し算1

```
__global__ void vecAdd(float* A, float* B, float* C) {
    int i = threadIdx.x;
    C[i] = A[i] + B[i];
}

int main(void) {
    vecAdd<<<1, N>>>(A, B, C);
}
```

並列の足し算2

```
__global__ void vecAdd(float* A, float* B, float* C) {
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    C[i] = A[i] + B[i];
}

int main(void) {
    vecAdd<<<Grid, Thread>>>(A, B, C);
}
```

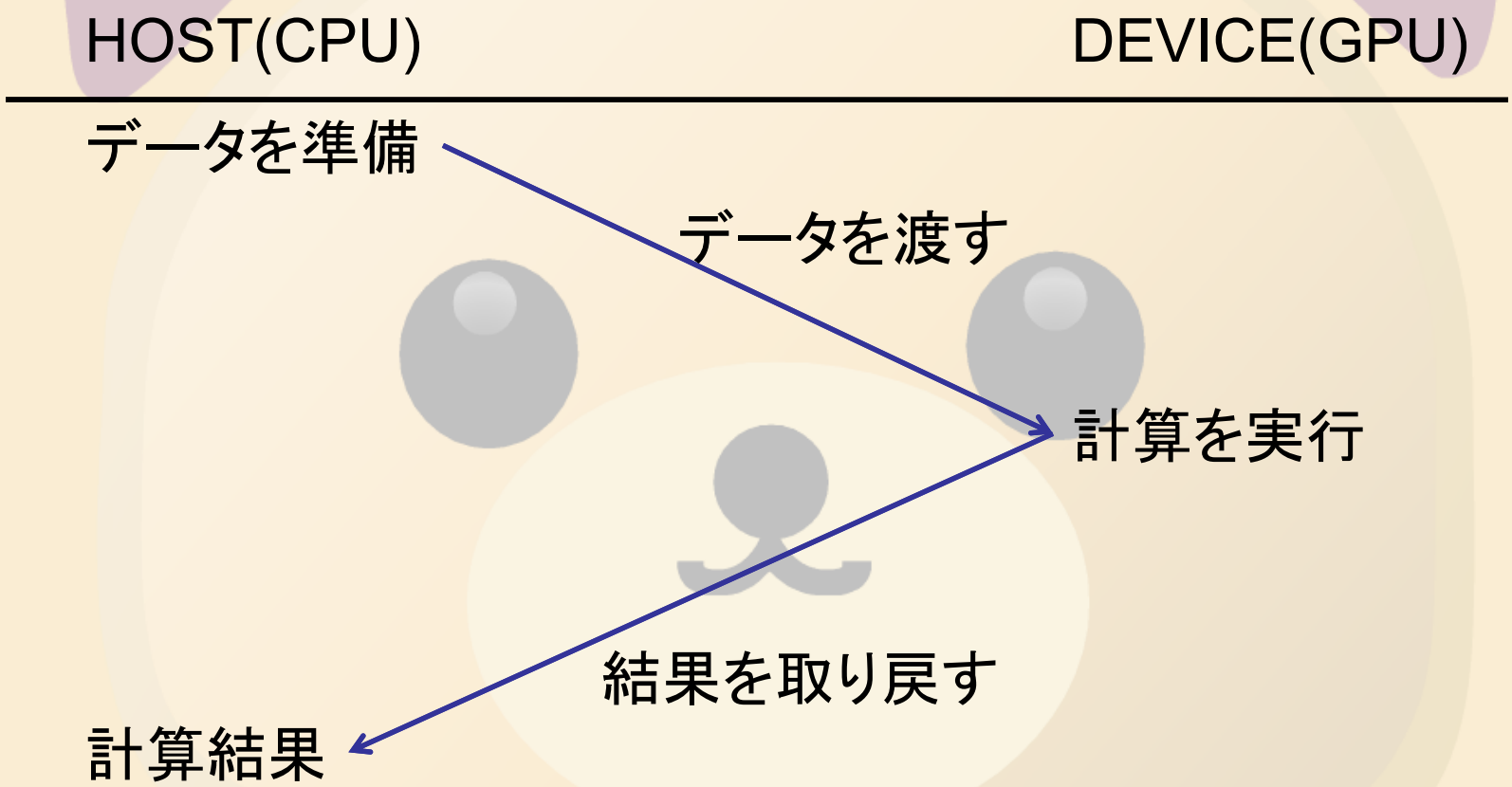
行列の足し算1

```
__global__ void matAdd(fload 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(void) {
    dim3 dimBlock(N, N);
    matAdd<<<1, dimBlock>>>(A, B, C);
}
```

行列の足し算2

```
__global__ void matAdd(fload A[N][N], float B[N][N], float C[N][N]) {
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    int j = blockIdx.y * blockDim.y + threadIdx.y;
    C[i][j] = A[i][j] + B[i][j];
}

int main(void) {
    dim3 dimBlock(16, 16);
    dim3 dimGrid((N + dimBlock.x - 1) / dimBlock.x,
                 (N + dimBlock.y - 1) / dimBlock.y));
    matAdd<<<dimGrid, dimBlock>>>(A, B, C);
}
```

実際には？



実際のコードにしてみると？

```
/* 実際のコード */
#include <stdio.h>
#include <cutil.h>

#define N 16

__global__ void vecAdd(float* A, float* B, float* C) {
    int i = threadIdx.x;
    C[i] = A[i] + B[i];
}

int main(int argc, char** argv) {
    CUT_DEVICE_INIT(argc, argv);
    float h_A[N], h_B[N], *h_C;
    float *d_A, *d_B, *d_C;
    int i;
```

実際のコードその2

```
for(i = 0; i < N; ++i) {
    h_A[i] = i;
    h_B[i] = i + 0.0001;
}    int mem_size = N*sizeof(float);

CUDA_SAFE_CALL(cudaMalloc((void**)&d_A, mem_size));
CUDA_SAFE_CALL(cudaMalloc((void**)&d_B, mem_size));
CUDA_SAFE_CALL(cudaMalloc((void**)&d_C, mem_size));

CUDA_SAFE_CALL(cudaMemcpy(d_A, h_A, mem_size,
                                    cudaMemcpyHostToDevice));
CUDA_SAFE_CALL(cudaMemcpy(d_B, h_B, mem_size,
                                    cudaMemcpyHostToDevice));

int grid = 1, thread = N;

vecAdd<<<grid, thread>>>(d_A, d_B, d_C);

h_C = (float*)malloc(mem_size);
```

実際のコードその3

```
    CUDA_SAFE_CALL(cudaMemcpy(h_C, d_C, mem_size,
                                        cudaMemcpyDeviceToHost));
    for(i = 0; i < N; ++i) {
        printf("%f + %f = %f¥n", h_A[i], h_B[i], h_C[i]);
    }
    free(h_C);
    CUDA_SAFE_CALL(cudaFree(d_A));
    CUDA_SAFE_CALL(cudaFree(d_B));
    CUDA_SAFE_CALL(cudaFree(d_C));
    CUT_EXIT(argc, argv);
}
```

な、長い…、ぜぇはぁ

Visual Studio 2005/2008の場合

C:¥Documents and Settings¥All Users
¥Application Data¥NVIDIA Corporation
¥NVIDIA CUDA SDK¥projects¥template

をコピーしてファイル内の「template」文字列を
自分の望むソリューション名に変更

Appendix

- http://ja.wikipedia.org/wiki/GPGPU
- http://ja.wikipedia.org/wiki/CUDA
- http://www.nvidia.co.jp/object/cuda_learn_jp.html