Bilgisayar Muhendisleri Odasi

Paralel Hesaplama

Paralel Hesaplama Nedir?

- Paralel hesaplama, basitce, tek islemci tarafindan hesaplanmasi cok fazla zaman gerektiren bir problemin, birden fazla islemciye paylastirilarak hesaplanmasidir.

Neden Paralel Hesaplama?

- Cunku bir bilgisayarin gorevi, insanlardan daha hizli hesaplama yapmaktir.

Paralel Hesaplaminin Gecmisi

- 1955: IBM ilk ticari floating-point mimariyi duyuruyor
- 1962: Burroughs, MIMD islemciyi (4 islemciye kadar) duyuruyor.
- 1971: Intel 4004 serisini piyasaya surmeye basliyor (ilk single-chip CPU)
- 1984: CRAY, 4 islemcili bilgisayarlar uretmeye basliyor.
- 1990lar: Tek islemcili makinalar liderliklerini surduruyor ve paralel hesaplamaya uygun makinalar sadece super bilgisayarlar olarak gozukuyor.
- 2002: Teknoloji, teorik limite erismeye basliyor ve islemcilerin performanslarini arttirmak zorlasiyor, ureticiler multi-processor calismalarina yogunlasiyor.

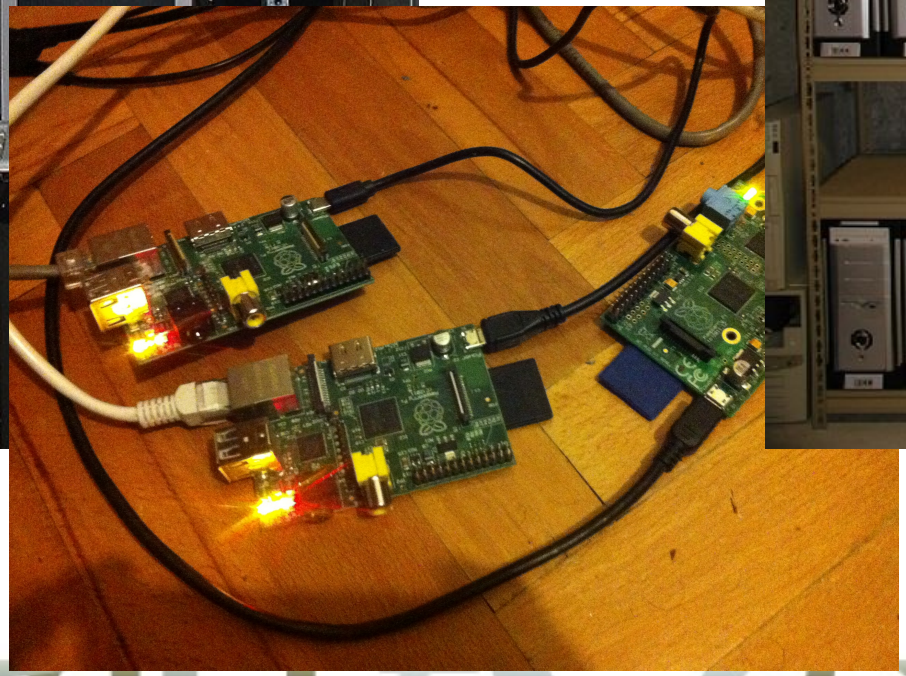
Paralel Hesaplama Icin Kullanilan Araclar

- Cok cekirdekli islemciler
- Clusterlar
- Super Bilgisayarlar
- Gonullu hesaplama yontemleri
- Ekran kartlari
- Bulut hesaplama cozumleri

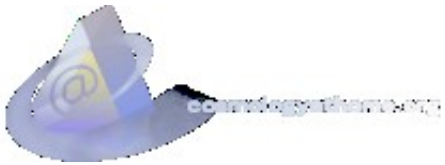
Cok Cekirdekli Islemciler

Super Bilgisayarlar

Cluster



Gonullu Hesaplama Yontemleri



Free Rainbow Tables
Distributed Rainbow Table Project

Ekran Kartlari

RADEON
GRAPHICS
AMD

Ekran Kartlari

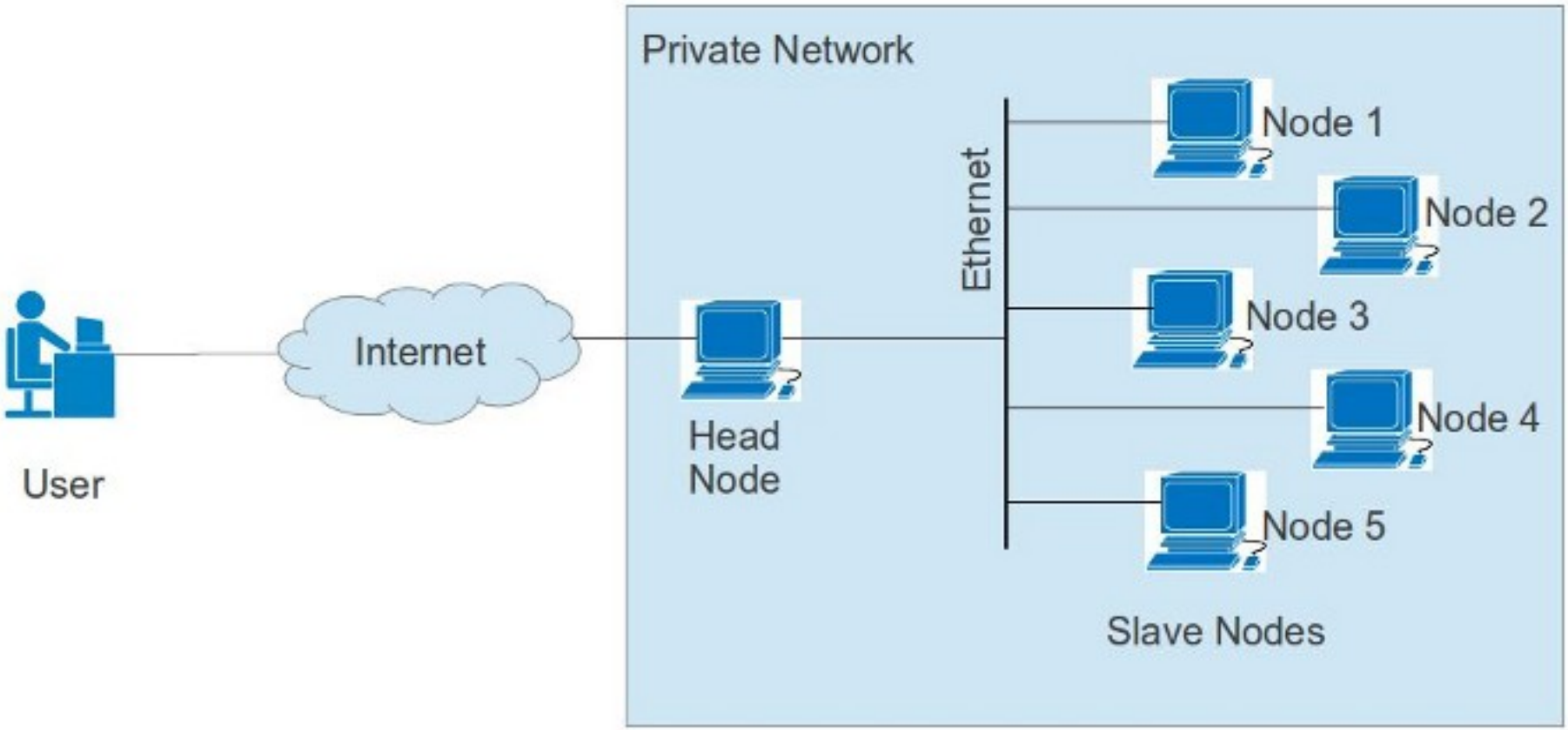
Bulut Cozumler

Cay, Simit...

MPI

- Message Passing Interface (MPI), birbirleri arasinda mesaj alisverisi yapan uygulamalar gelistirmek icin bir standarttir.
- OpenMPI, MPICH...

MPI


Private Network
Node 1
Node 2
Node 3
Node 4
Node 5
Ethernet
Internet
Head
Node
User
Slave Nodes

MPI

```
/* C Example */
#include <stdio.h>
#include <mpi.h>

int main (int argc, char **argv)
{
    int rank, size;
    MPI_Init (&argc, &argv);    /* starts MPI */
    MPI_Comm_rank (MPI_COMM_WORLD, &rank);          /* get current process id */
    MPI_Comm_size (MPI_COMM_WORLD, &size);/* get number of processes */
    printf( "Hello world from process %d of %d\n", rank, size );
    MPI_Finalize();
    return 0;
}
```

MPI

```
/* C Example */
#include <stdio.h>
#include <mpi.h>

int main (int argc, char **argv)
{
    int rank, size;
    MPI_Init (&argc, &argv);    /* starts MPI */
    MPI_Comm_rank (MPI_COMM_WORLD, &rank);          /* get current process id */
    MPI_Comm_size (MPI_COMM_WORLD, &size);/* get number of processes */
    if( rank == 0)
    {
            printf("Hello, I am the head node and my rank is %d. Total number of processors is
%d\n", rank, size);
     }
    else
    {
             printf( "Hello, I am slave node %d\n", rank);
     }
    MPI_Finalize();
    return 0;
}
```

MPI

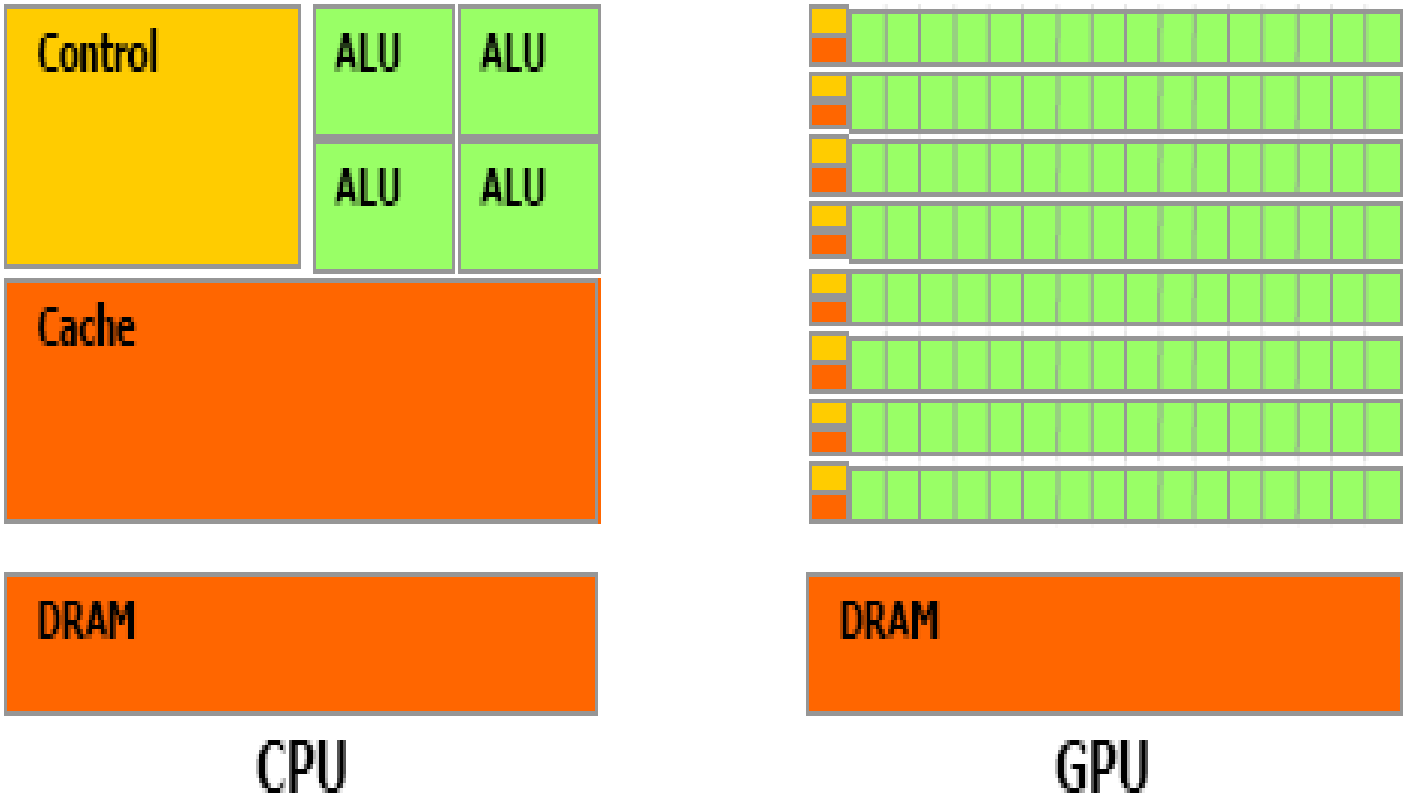
- int MPI_Send(void *buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm);

- int MPI_Recv(void *buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status);

MPI

Basit bir MPI uygulamasi ornegi

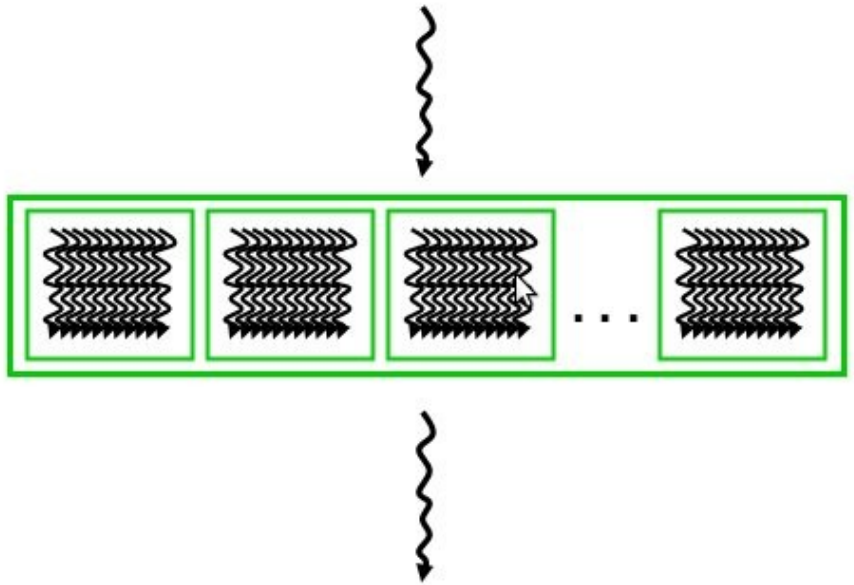
CUDA



CUDA

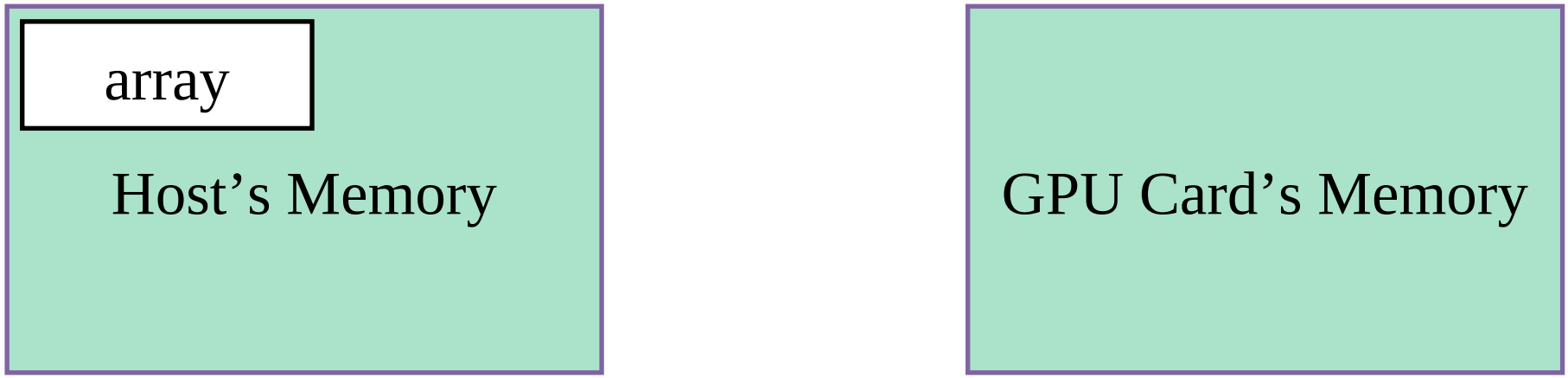
Serial Code (Host)

Parallel Code
(Device)

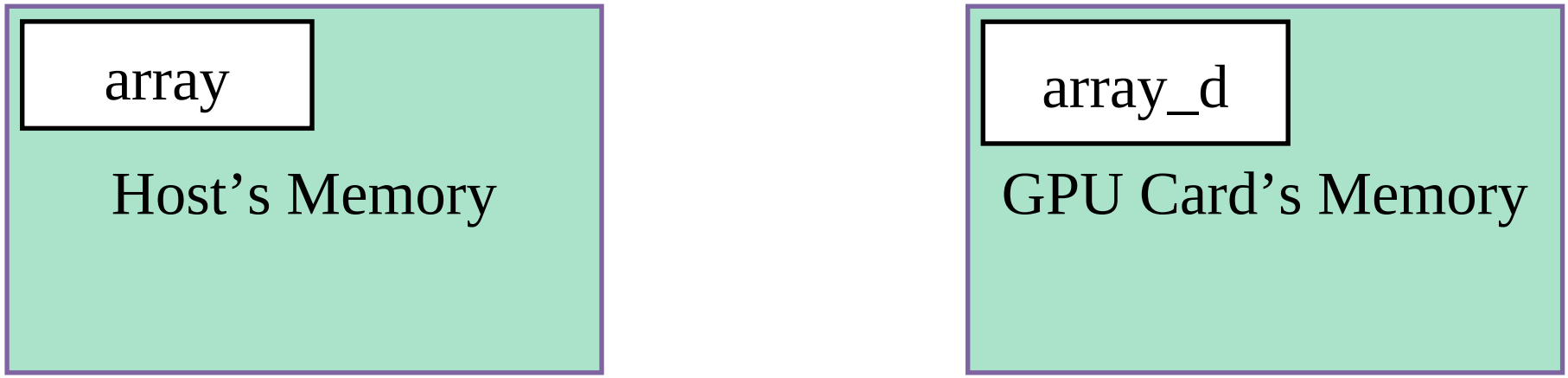


Serial Code (Host)

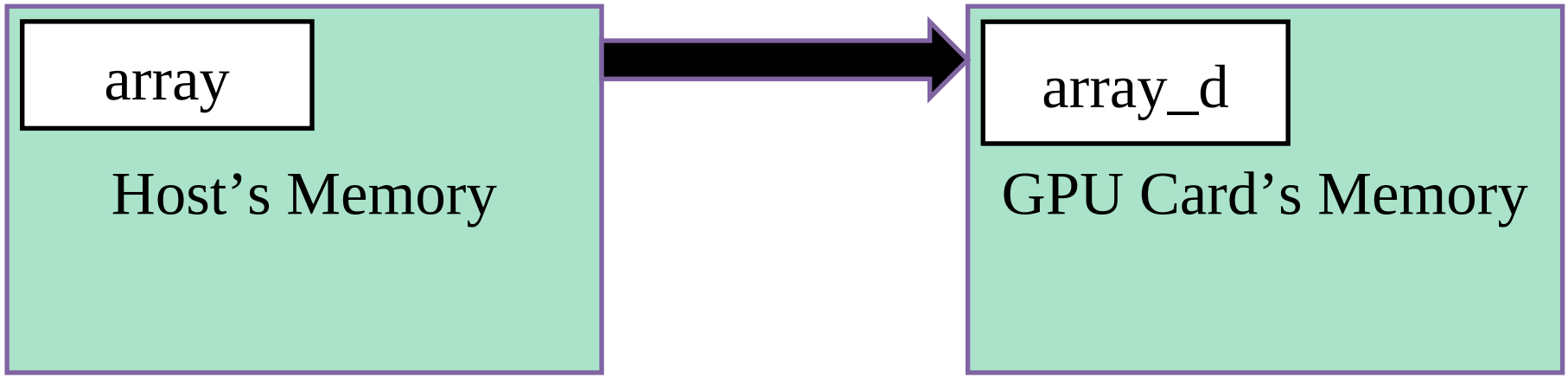
CUDA



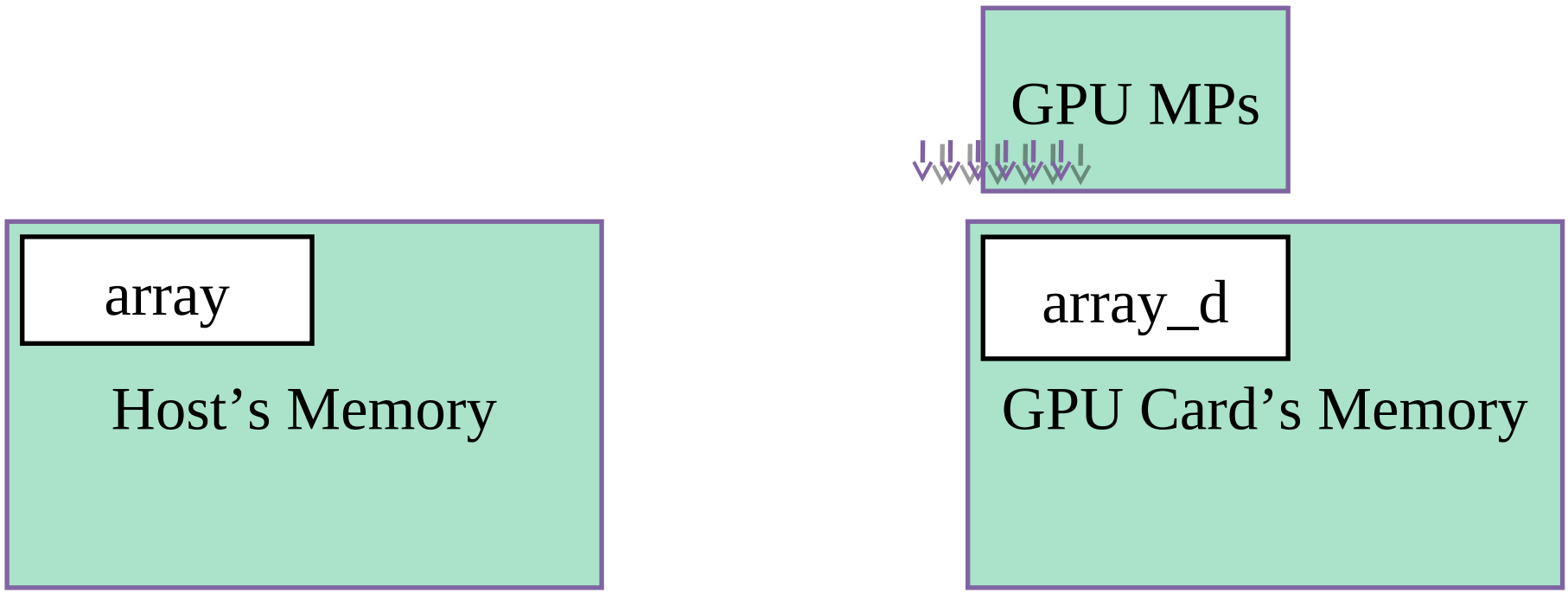
CUDA



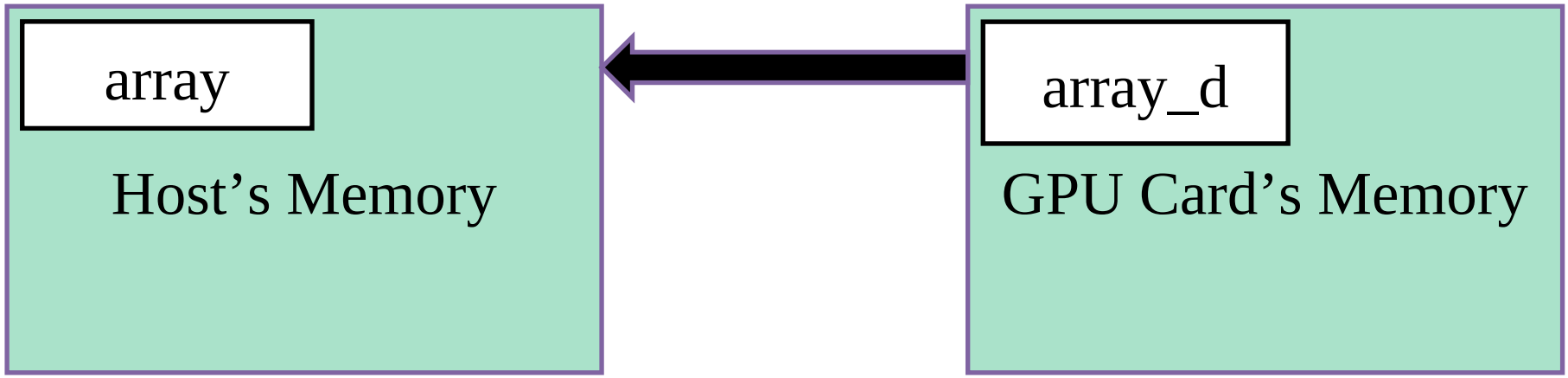
CUDA



CUDA



CUDA



CUDA

- **Type Qualifiers**
  - **global, device, shared, local, constant**
- **Keywords**
  - **threadIdx, blockIdx**
- **Intrinsics**
  - **__syncthreads**
- **Runtime API**
  - **Memory, symbol, execution management**
- **Function launch**

```
__device__ float filter[N];
__global__ void convolve (float *image)  {
__shared__ float region[M];

  ...

  region[threadIdx] = image[i];

  __syncthreads()

  ...
  image[j] = result;

// Allocate GPU memory
void *myimage = cudaMalloc(bytes)
a
// 100 blocks, 10 threads per block
convolve<<<100, 10>>> (myimage);
```

CUDA

	Executed on the:	Only callable from the:
`__device__ float DeviceFunc()`	device	device
`__global__ void KernelFunc()`	device	host
`__host__ float HostFunc()`	host	host

CUDA

Matrix multiplication serial code example for size NxN;

```
for(int i=0; i<N; i++) {
    for(int j=0; j<N; j++) {
        for(int k=0; k<N; k++) {
            c[i][j] += a[i][k] * b[k][j];
        }
    }
}
```

CUDA

Matrix multiplication parallel code example for size NxN;

```
int i = threadIdx.x;
int j = threadIdx.y;
int sum = 0;

for(int k=0; k<N; k++)
{
    sum += a[i*N+k] * b[k*N+j];
}
c[i*N+j] = sum;
```

CUDA

Basit bir CUDA programı

Tesekkurler...