CUDA ON ARM PLATFORM—基于ARM平台的JETSON NANO存储单元调用

NVIDIA企业级开发者社区 何琨

基于ARM平台的JETSON NANO 存储单元调用

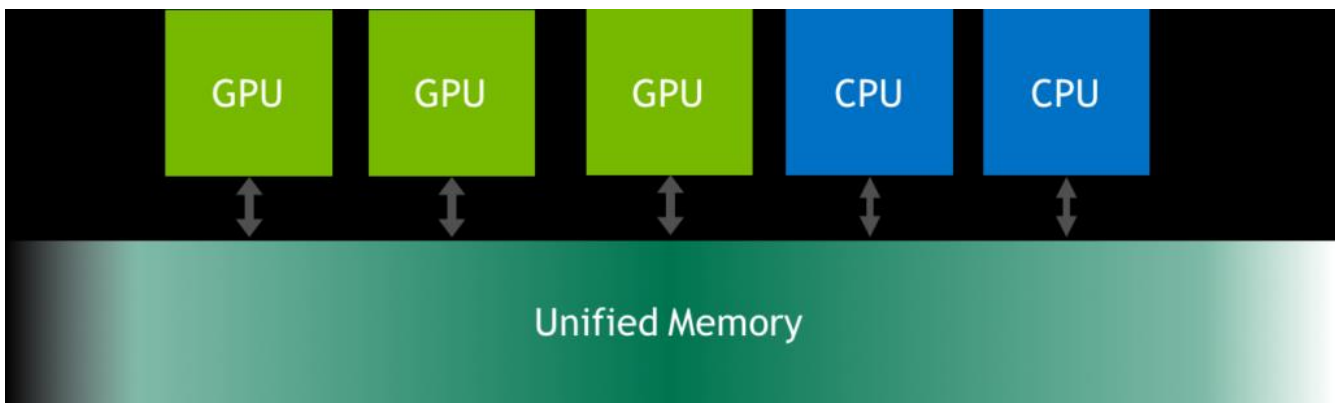
AGENDA

- 统一内存的基本概念
- 基于ARM平台的Jetson nano的存储单元特点
- 如何更有效的利用Jetson的存储单元

统一内存的基本概念

Unified Memory:

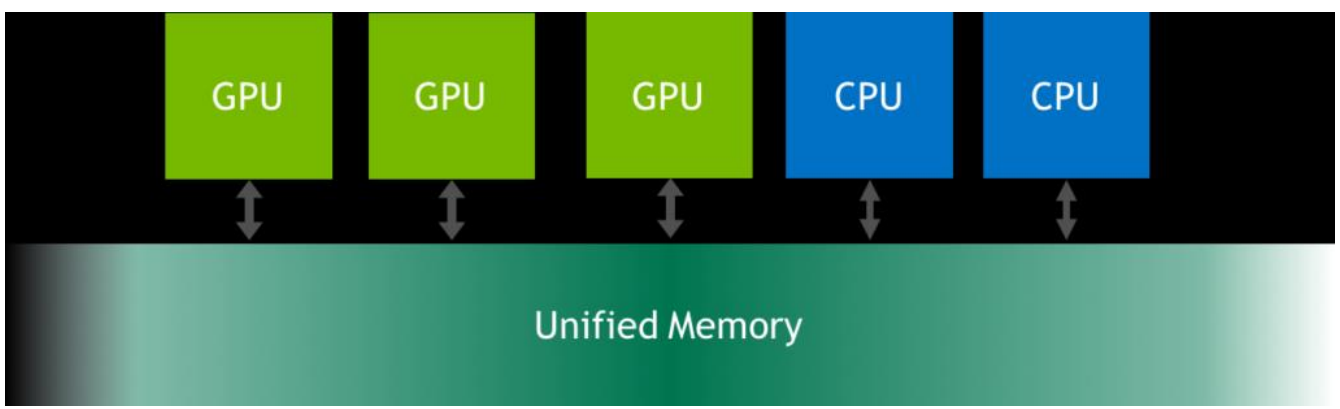
- 统一内存是可从系统中的任何处理器访问的单个内存地址空间。这种硬件/软件技术允许应用程序分配可以从 CPU s 或 GPUs 上运行的代码读取或写入的数据。分配统一内存非常简单，只需将对 malloc() 或 new 的调用替换为对 cudaMallocManaged() 的调用，这是一个分配函数，返回可从任何处理器访问的指针。



统一内存的基本概念

Unified Memory:

- a GPU with SM architecture 3.0 or higher (Kepler class or newer)
- a 64-bit host application and non-embedded operating system (Linux or Windows)



统一内存的基本概念

Unified Memory:

CPU Code

```
void sortfile(FILE *fp, int N) {
  char *data;
  data = (char *)malloc(N);

  fread(data, 1, N, fp);

  qsort(data, N, 1, compare);


  use_data(data);

  free(data);
}
```

CUDA 6 Code with Unified Memory

```
void sortfile(FILE *fp, int N) {
  char *data;
  cudaMallocManaged(&data, N);

  fread(data, 1, N, fp);

  qsort<<<...>>>(data,N,1,compare);
  cudaDeviceSynchronize();

  use_data(data);

  cudaFree(data);
}
```

统一内存的基本概念

Unified Memory: 两种实现方法

1. cudaError_t cudaMallocManaged(void **devPtr, size_t size, unsigned int flags=0);

2. __managed__

统一内存的基本概念

Unified Memory: 两种实现方法

1. cudaError_t cudaMallocManaged(void **devPtr, size_t size, unsigned int flags=0);

```
__global__ void printme(char *str) {
    printf(str);
}
int main() {

    char *s;
    cudaMallocManaged(&s, 100);
    strncpy(s, "Hello Unified Memory\n", 99);
    printme<<< 1, 1 >>>(s);
    cudaDeviceSynchronize();
    cudaFree(s);
    return  0;
}
```

统一内存的基本概念

Unified Memory: 两种实现方法

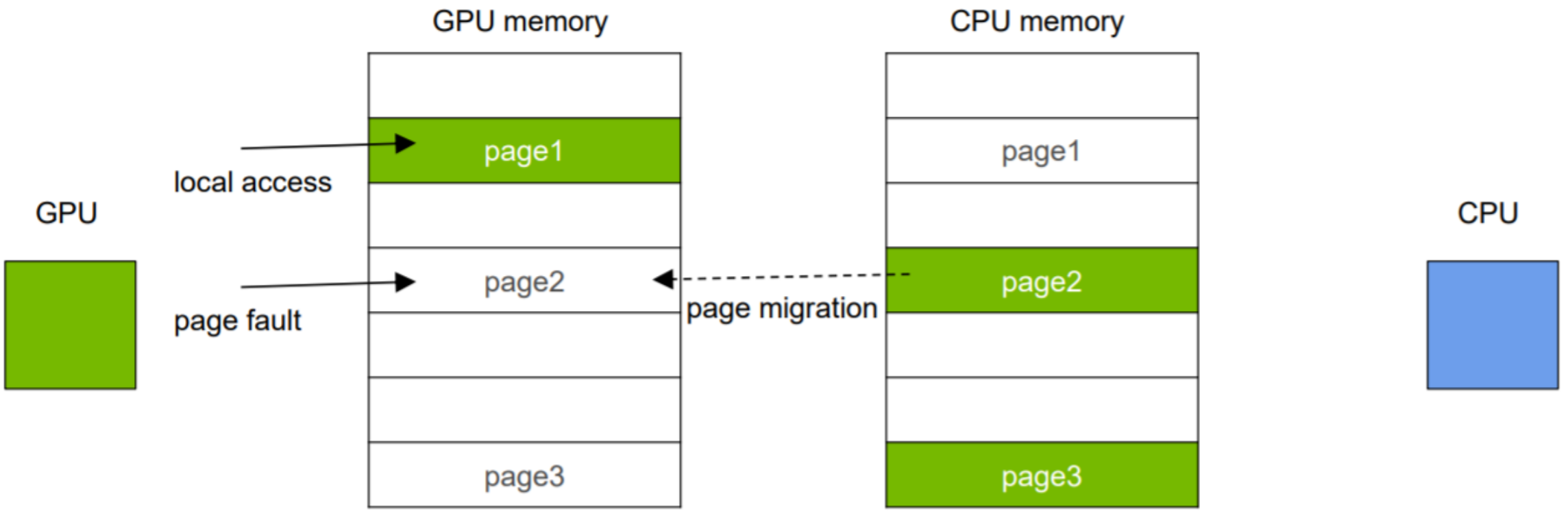
2. __managed__

```
__device__ __managed__ int x[2];
__device__ __managed__ int y;
__global__ void kernel() {
    x[1] = x[0] + y;
}
int main() {
    x[0] = 3;
    y = 5;
    kernel<<< 1, 1 >>>();
    cudaDeviceSynchronize();
    printf("result = %d\n", x[1]);
    return  0;
}
```

File-scope and global-scope CUDA __device__ variables may also opt-in to Unified Memory management by adding a new __managed__ annotation to the declaration. These may then be referenced directly from either host or device code

统一内存的基本概念

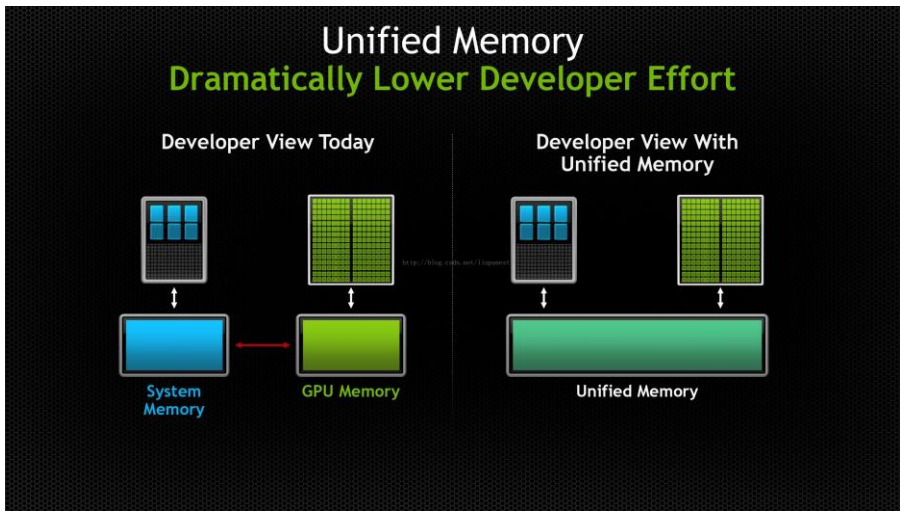
Unified Memory:

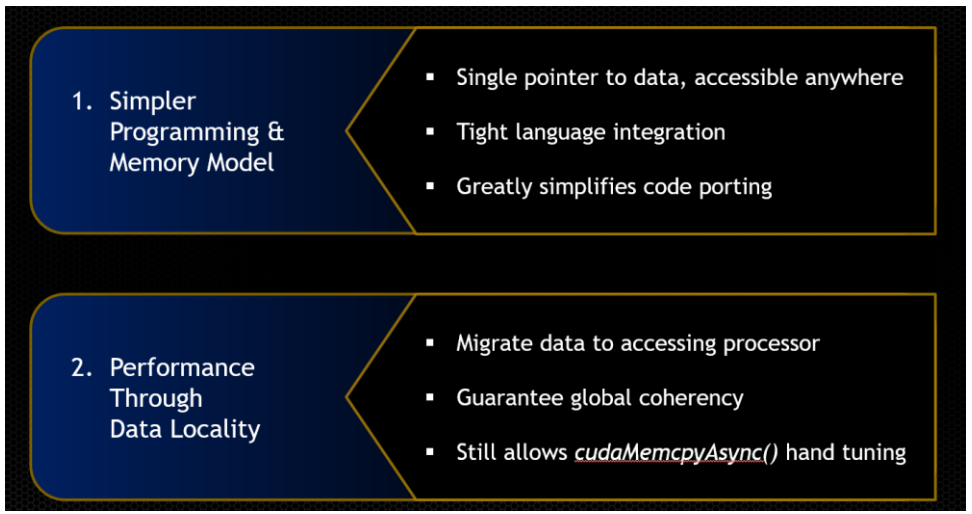


统一内存的基本概念

Unified Memory:

- 可直接访问CPU内存、GPU显存，不需要手动拷贝数据。
- CUDA 在现有的内存池结构上增加了一个统一内存系统，程序员可以直接访问任何内存/显存资源，或者在合法的内存空间内寻址，而不用管涉及到的到底是内存还是显存。
- CUDA 的数据拷贝由程序员的手动转移，变成自动执行，因此，它仍然受制于PCI-E的带宽和延迟。



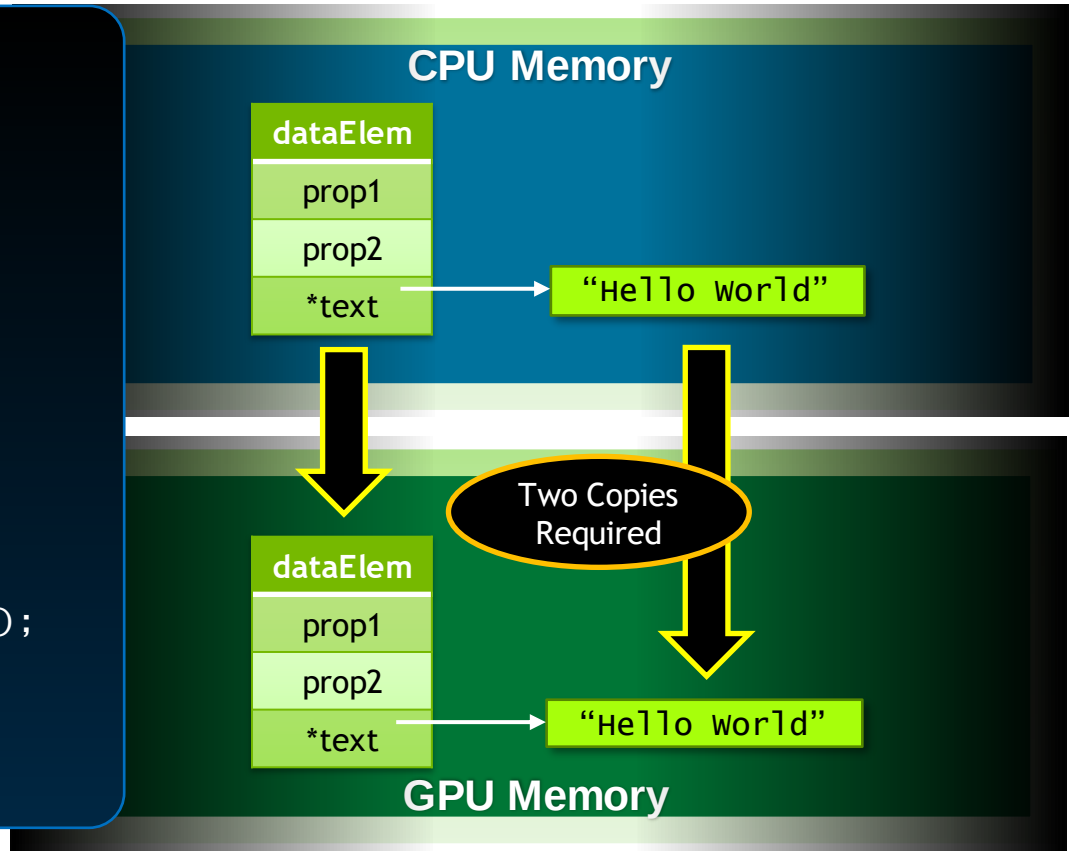


统一内存的基本概念

Unified Memory :

案例

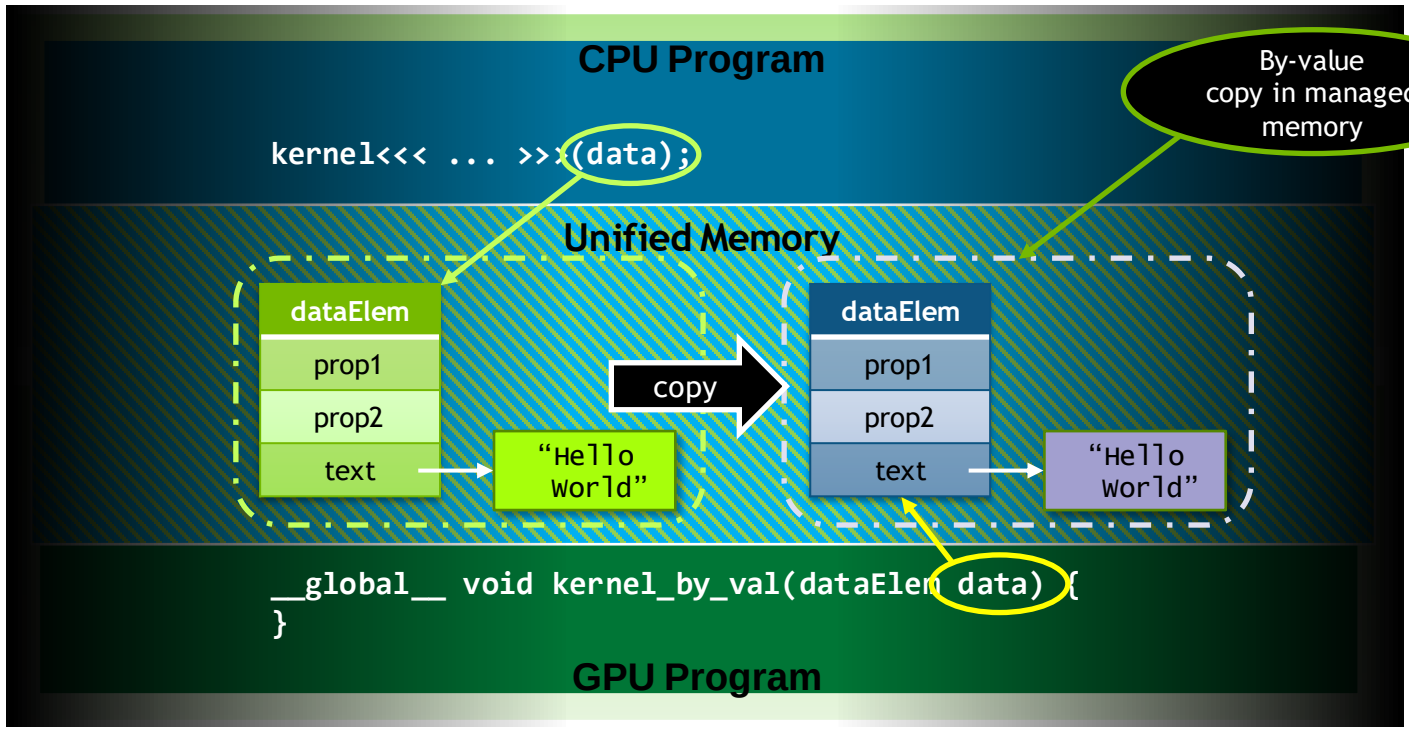
```
void launch(dataElem *elem) {
    dataElem *g_elem;
    char *g_text;

    int textlen = strlen(elem->text);

    // Allocate storage for struct and text
    cudaMalloc(&g_elem, sizeof(dataElem));
    cudaMalloc(&g_text, textlen);

    // Copy up each piece separately, including
    // new "text" pointer value
    cudaMemcpy(g_elem, elem, sizeof(dataElem));
    cudaMemcpy(g_text, elem->text, textlen);
    cudaMemcpy(&(g_elem->text), &g_text,
                                sizeof(g_text));

    // Finally we can launch our kernel, but
    // CPU & GPU use different copies of "elem"
    kernel<<< ... >>>(g_elem);
}
```



统一内存的基本概念

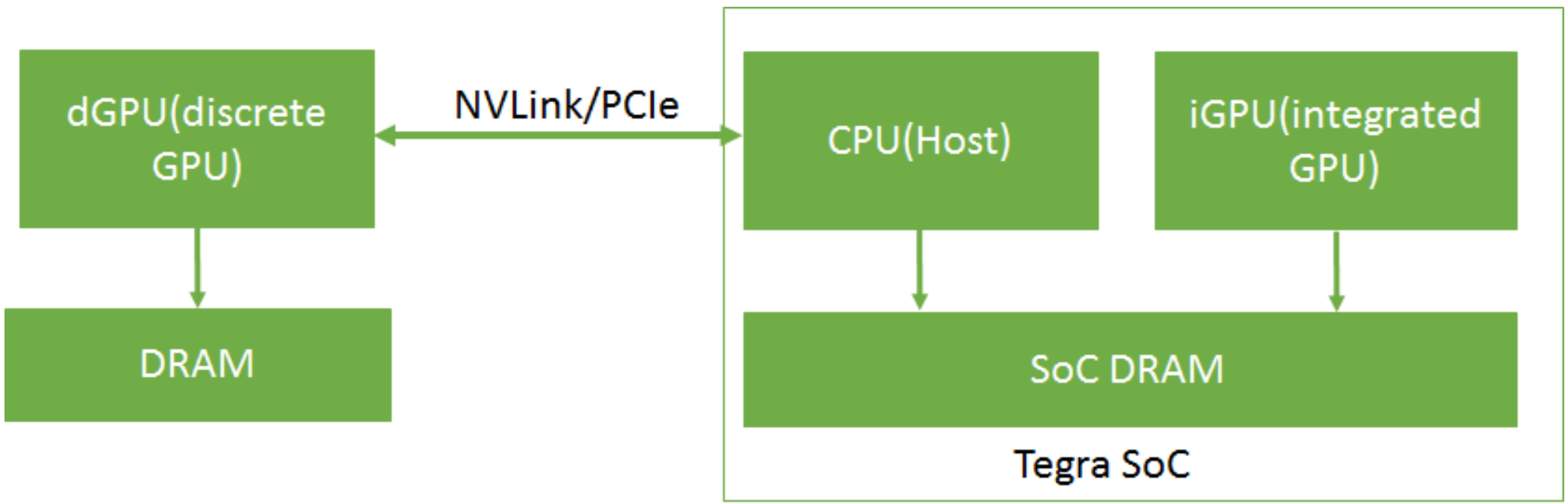
Unified Memory :

案例



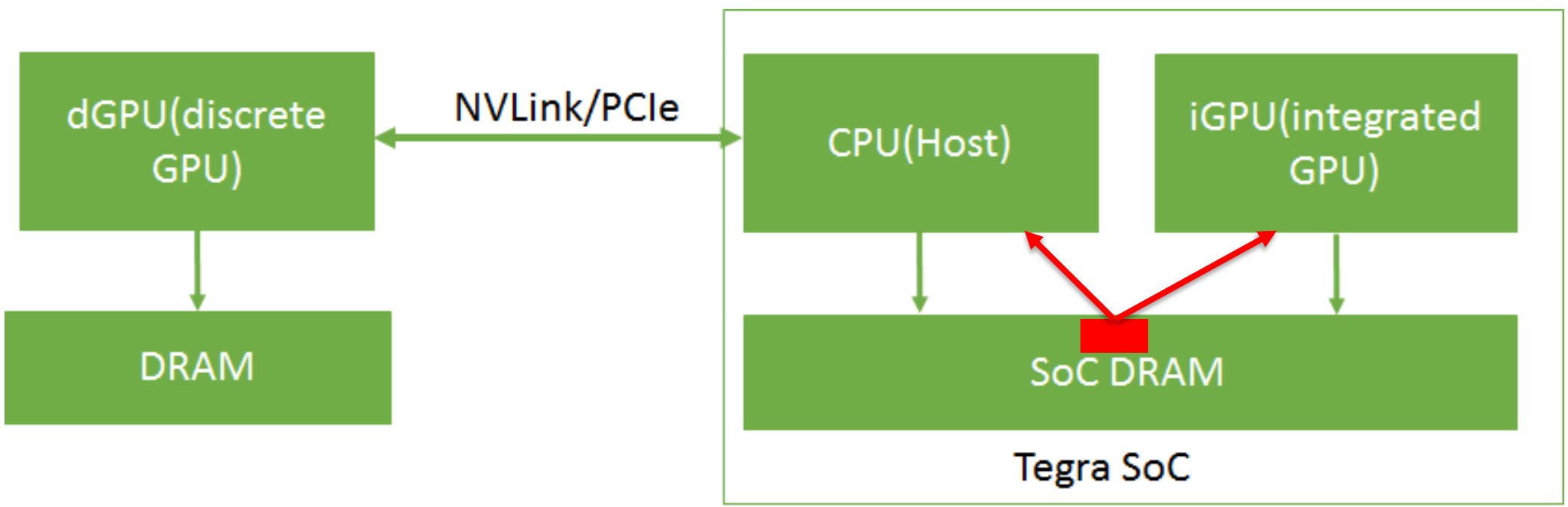

```
// Deriving from "Managed" allows
pass-by-reference
class String : public Managed {
    int length;
    char *data;

    // Unified memory copy
constructor allows pass-by-value
    String (const String &s) {
        length = s.length;
        cudaMallocManaged(&data,
length);
        memcpy(data, s.data,
length);
    }
};
```

基于ARM平台的JETSON NANO的存储单元特点

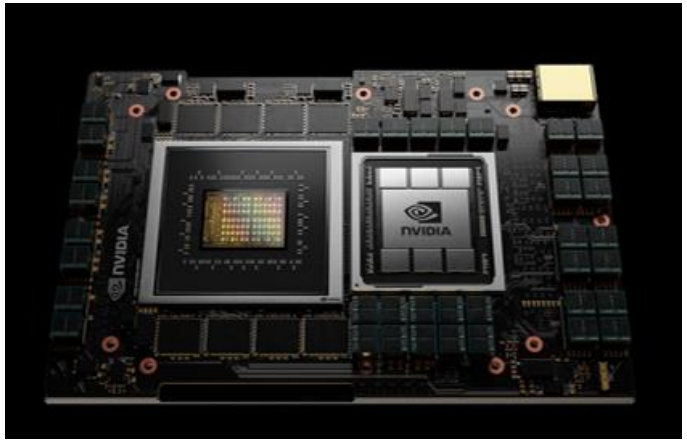


基于ARM平台的JETSON NANO的存储单元特点

Because device memory, host memory, and unified memory are allocated on the same physical SoC DRAM, duplicate memory allocations and data transfers can be avoided



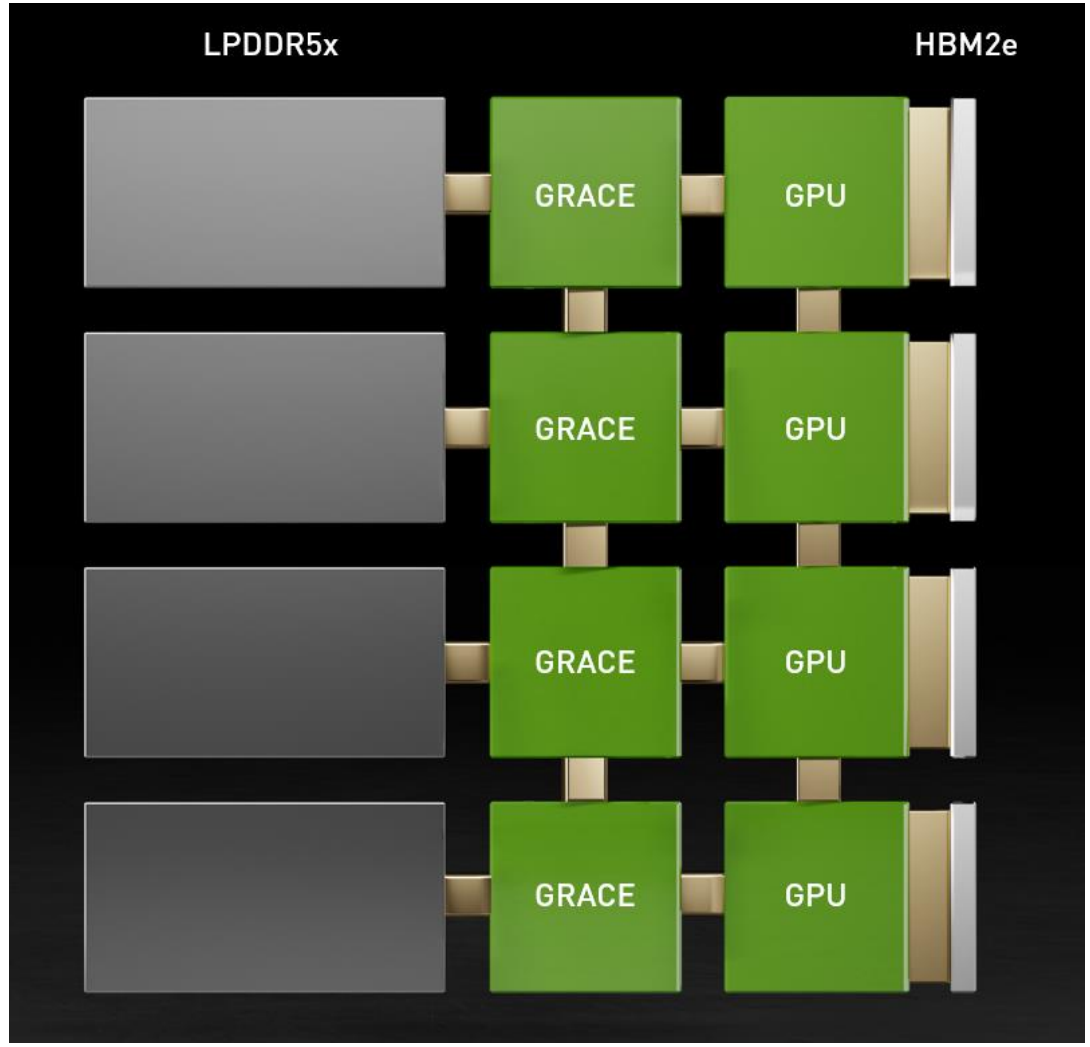
将来!



A NEW COMPUTING ARCHITECTURE FOR AI AND DATA SCIENCE

30X Increase System Memory to GPU

GPU	8,000	GB/sec	
CPU	500	GB/sec	
NVLINK	500	GB/sec	
Mem-to-GPU	2,000	GB/sec	30X



存储单元分配

由于CPU和iGPU可以直接访问这两种内存，所以可以使用固定内存或统一内存来减少CPU和iGPU之间的数据传输开销。

Table 1. Characteristics of Different Memory Types in a Tegra System

Memory Type	CPU	iGPU	Tegra®-connected dGPU
Device memory	Not directly accessible	Cached	Cached
Pageable host memory	Cached	Not directly accessible	Not directly accessible
Pinned host memory	Uncached where compute capability is less than 7.2. Cached where compute capability is greater than or equal to 7.2.	Uncached	Uncached
Unified memory	Cached	Cached	Not supported

更多资源:

https://developer.nvidia-china.com

何琨-Ken

北京 密云

扫一扫上面的二维码图案，加我微信

https://www.nvidia.cn/developer/community-training/


NVIDIA