Optimizing Host-Device Data Communication I - *Pinned Host Memory*

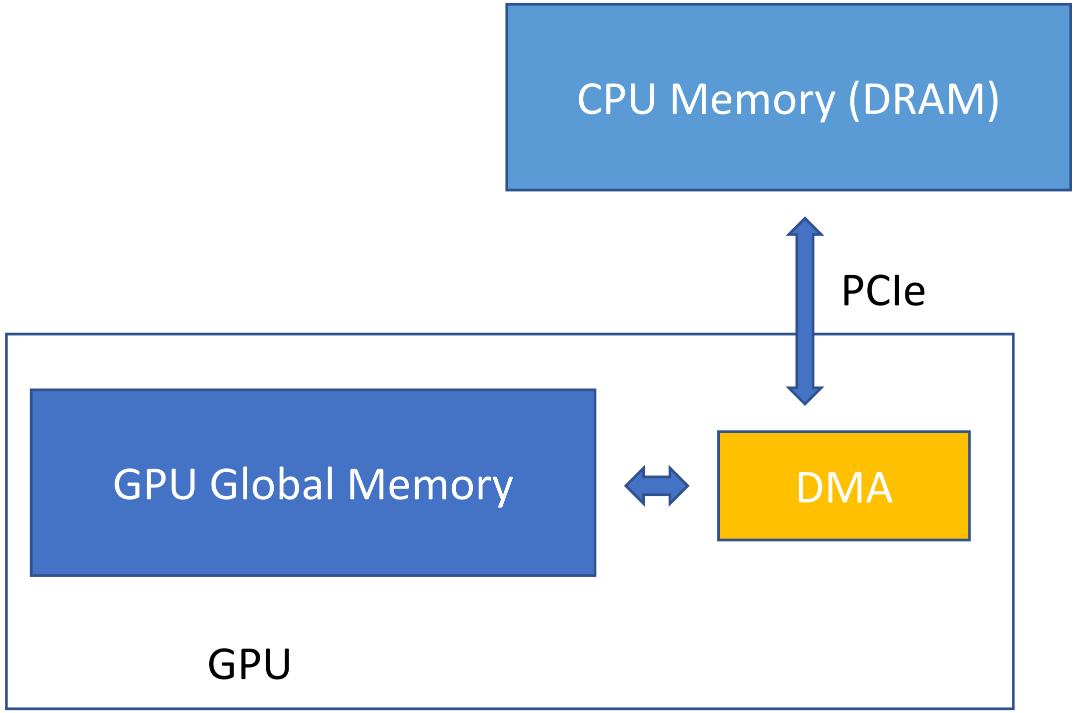
Stefano Markidis

Four Key-Points

1. Communication between the host and device are the slowest link of data movement involved in GPU computing, so we optimize transfers
2. Pinned host memory allows us to avoid intermediate transfers
3. For allocating pinned memory, instead of using malloc we use `cudaHostAlloc`
4. When using pinned memory, we could batch all the small transfers in one large data transfer

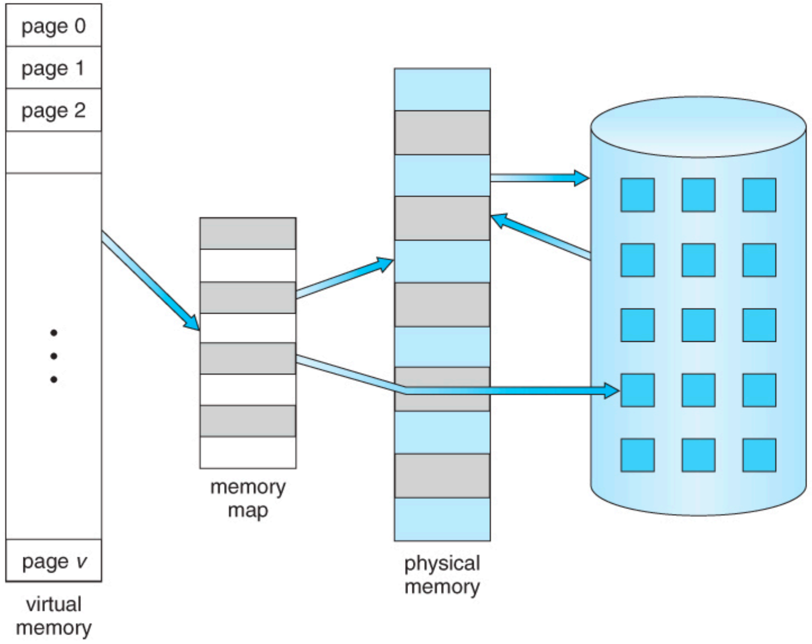
CPU-GPU Data Communication with DMA

- DMA (Direct Memory Access) hardware is used for `cudaMemcpy()` for increased efficiency
  - Frees CPU for other tasks
  - Transfers a number of bytes requested by OS
  - Uses system interconnect, typically PCIe / NVLink



Virtual Memory Management

- Hosts uses virtual memory management
- Many virtual memory spaces mapped into a single physical memory
    - Virtual addresses (pointer values) are translated into physical addresses
    - Not all variables and data structures are always in the physical memory
- Each virtual address space is divided into pages when mapped into physical memory
- Memory pages can be paged out to make room
- If a variable is in the physical memory is checked at address translation time
- Whether a variable is in the physical memory is checked at address translation time



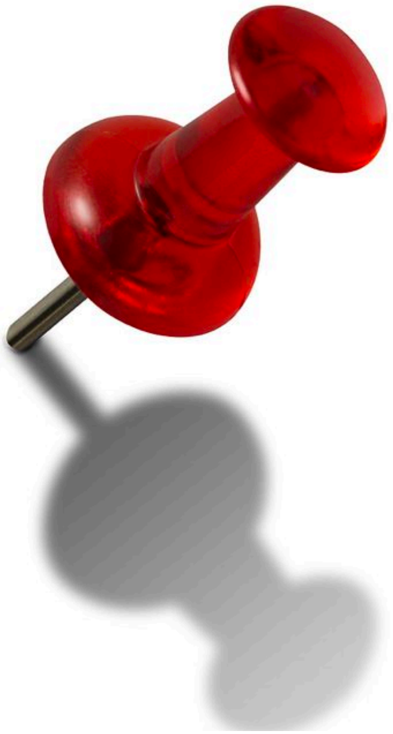
https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/9_VirtualMemory.html

Data Transfer and Virtual Memory

- DMA uses physical addresses
  - When `cudaMemcpy()` copies an array, it is implemented as one or more DMA transfers
  - Address is translated and page presence checked at the beginning of each DMA transfer
  - No address translation for the rest of the same DMA transfer so that high efficiency can be achieved
- The OS could accidentally page-out the data that is being read or written by a DMA and page-in another virtual page into the same physical location

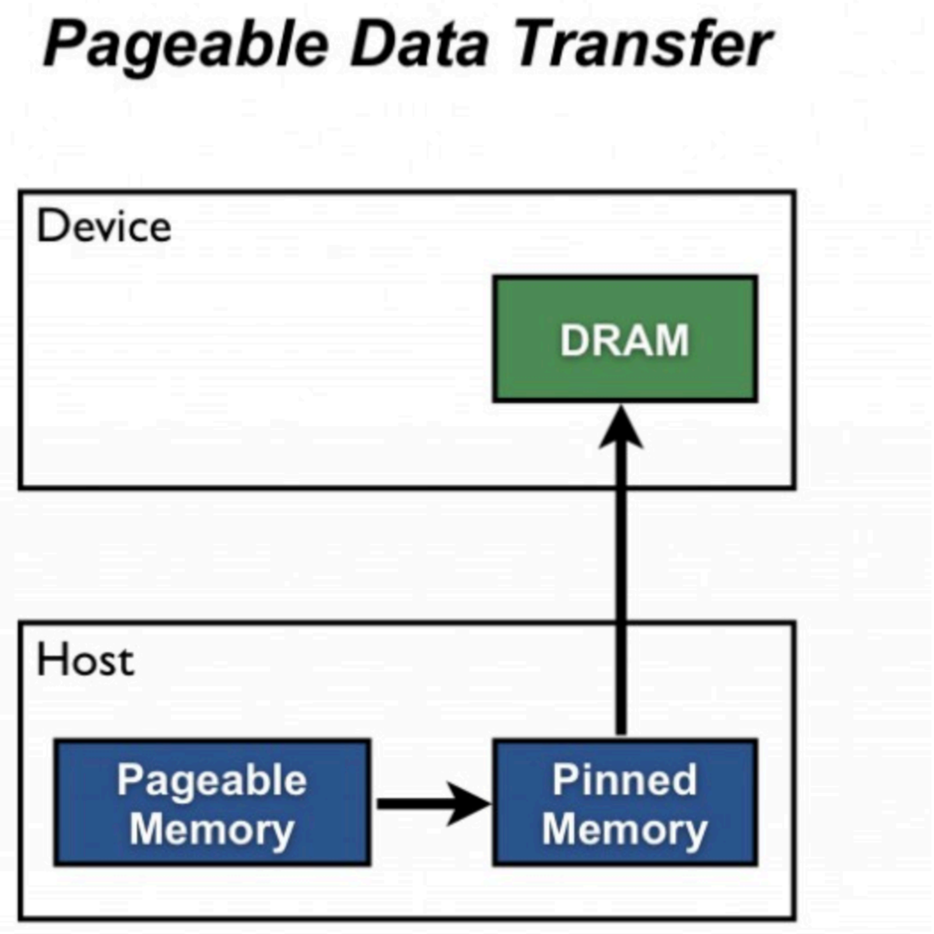
Pinned Memory and DMA Data Transfer

- Pinned memory are virtual memory pages that are specially marked so that they cannot be paged out
- Allocated with a special system API function call
  - *Page Locked Memory*, *Locked Pages*, etc.
- CPU memory that serve as the source of destination of a DMA transfer must be allocated as pinned memory

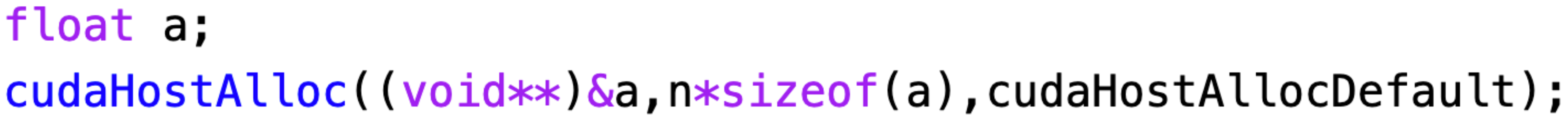


CUDA Data Transfer uses Pinned Memory

- If a source or destination of a `cudaMemcpy()` in the host memory is not allocated in pinned memory, it needs to be first copied to a pinned memory → **extra overhead**
- `cudaMemcpy()` is faster if the host memory source or destination is allocated in pinned memory since no extra copy is needed



Allocate/Free Pinned Memory



```
float a;
cudaHostAlloc((void**)&a,n*sizeof(a),cudaHostAllocDefault);
```

- `cudaHostAlloc()` takes 3 parameters:
  1. Address of pointer to the allocated memory
  2. Size of the allocated memory in bytes
  3. Option – use cudaHostAllocDefault for now
- `cudaFreeHost()` takes one parameter:
  - Pointer to the memory to be freed

Code Example

instead of `malloc()`

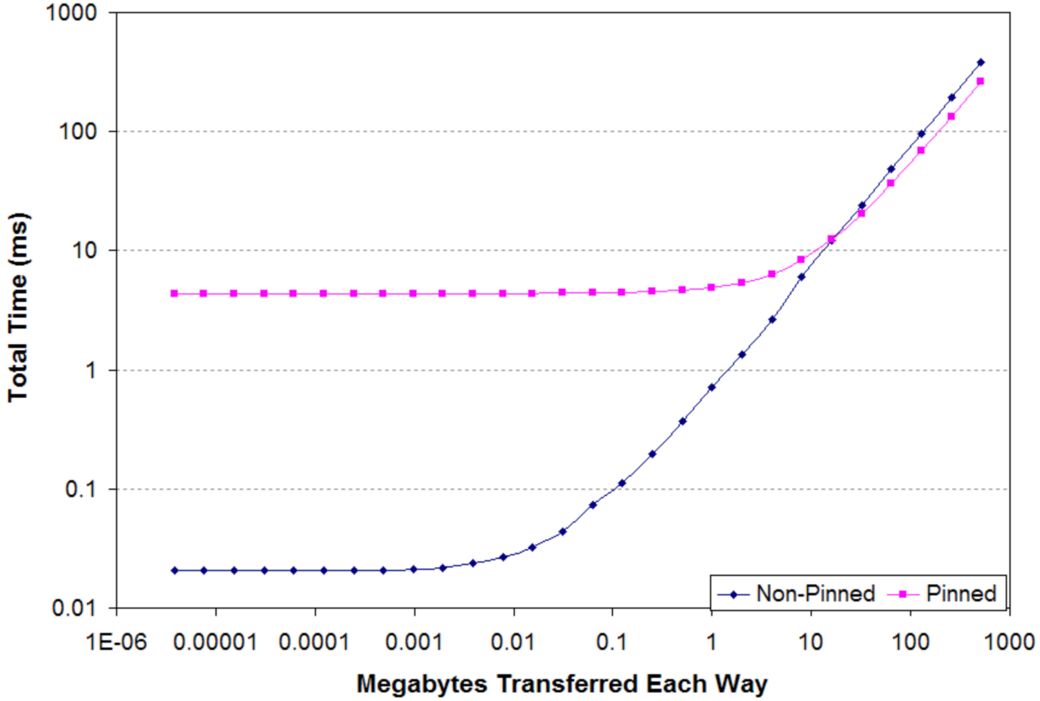
```
int main() {
   float *h_A, *h_B, *h_C; ...
   cudaHostAlloc((void **) &h_A, N* sizeof(float), cudaHostAllocDefault)
   cudaHostAlloc((void **) &h_B, N* sizeof(float), cudaHostAllocDefault)
   cudaHostAlloc((void **) &h_C, N* sizeof(float), cudaHostAllocDefault)
   ...
   vecAdd(h_A, h_B, h_C, N);
   ...
}
```

Using Pinned Memory in CUDA

- Use the allocated pinned memory and its pointer the same way as those returned by `malloc();`
- The only difference is that the **allocated memory cannot be paged by the OS**
- The `cudaMemcpy()` function should be faster with pinned memory but it depends …
- Pinned memory is a **limited resource**
  - It is possible for **pinned memory allocation to fail**
    - **we should check for errors**!

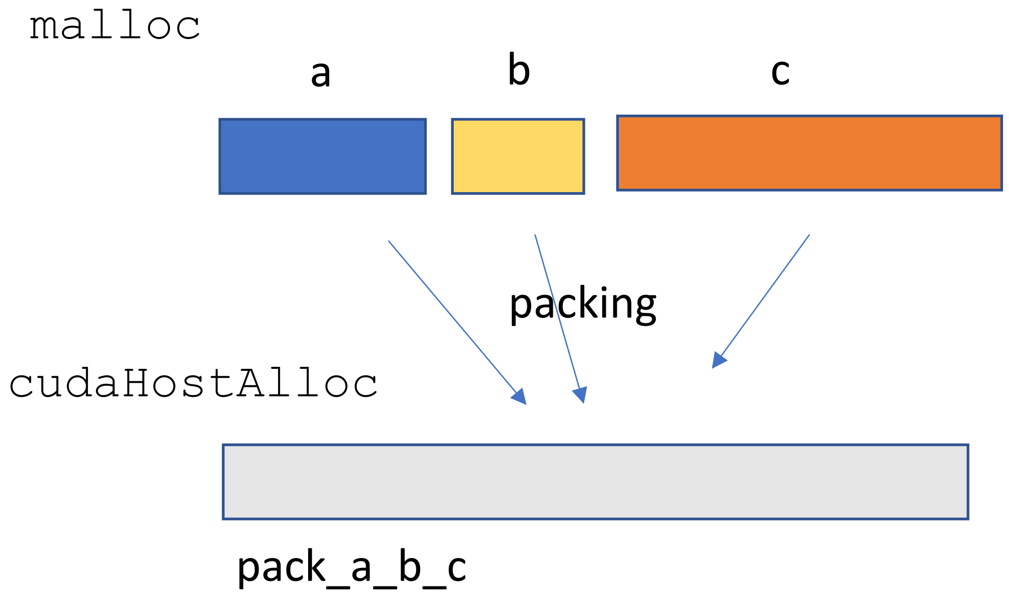
Performance Advantages

- Pinned memory is much more **expensive to allocate and deallocate** but provides higher transfer throughput for large memory transfers.
- The speed-up with pinned memory depends on device compute capability
- **Batching many smaller transfers into one larger transfer** improves performance of pinned memory.



Batching Small Size Transfers

- Due to the overhead associated with each transfer
  - it is preferable to batch many **small transfers** together into a single transfer
- We can use a **temporary pinned array** and pack it with the data to be transferred



To Summarize

- Transfers between the host and device are the slowest link of data movement so **we should optimize data transfers**.
- Pinned memory on the host allows us to avoid intermediate transfers and achieve higher data transfer rate (bandwidth)
- For allocating pinned memory, instead of using `malloc` we use `cudaHostAlloc` and `cudaMallocHost`
- When using pinned memory, we batch all the small transfers in one large data transfer to avoid the overhead of small data transfers.