Towards Automatic Heterogeneous Computing Performance Analysis

Carl Pearson
pearson@illinois.edu
Adviser: Wen-Mei Hwu

2018 03 30

Outline

High Performance Computing Challenges

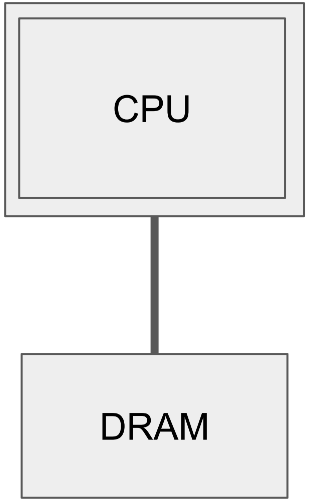
Vision

CUDA Allocation and Transfer Background

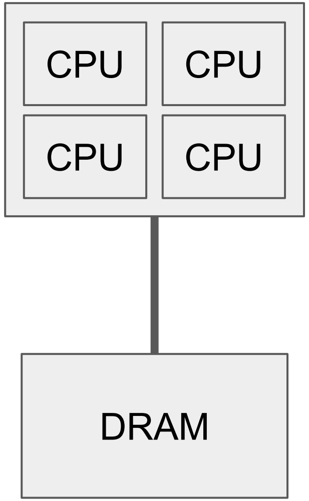
System Characterization

Software Characterization

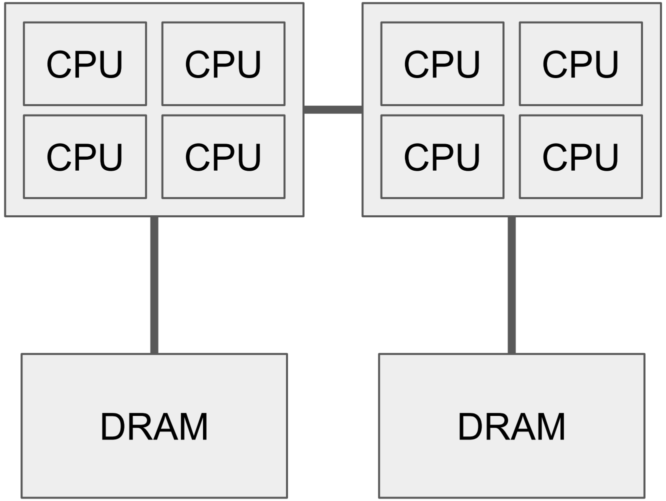
Out-of-Order execution
Vectorization
Cache effects



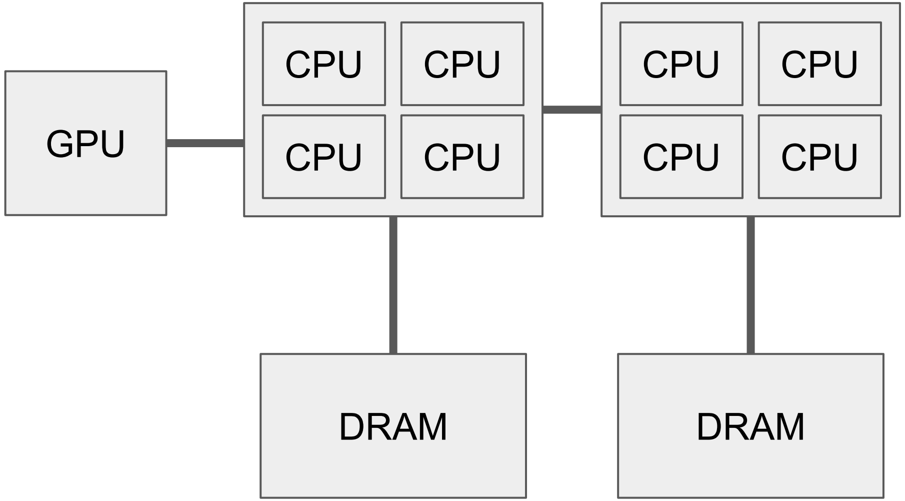
Out-of-Order execution
Vectorization
Cache effects
Multi-threading



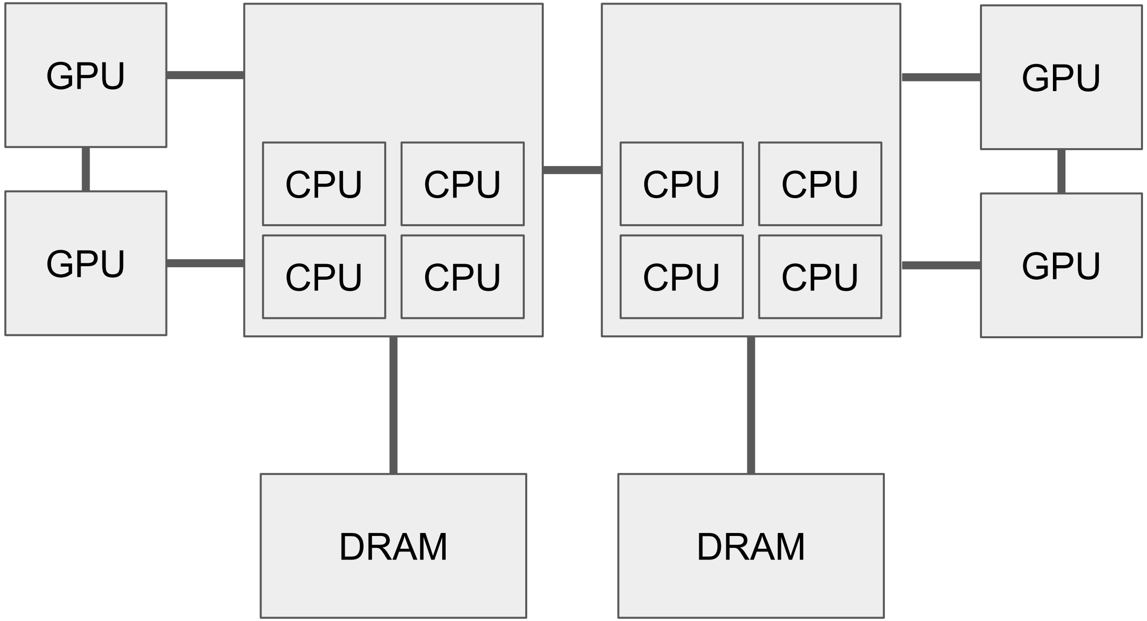
Out-of-Order execution
Vectorization
Cache effects
Multi-threading
Non-uniform memory access



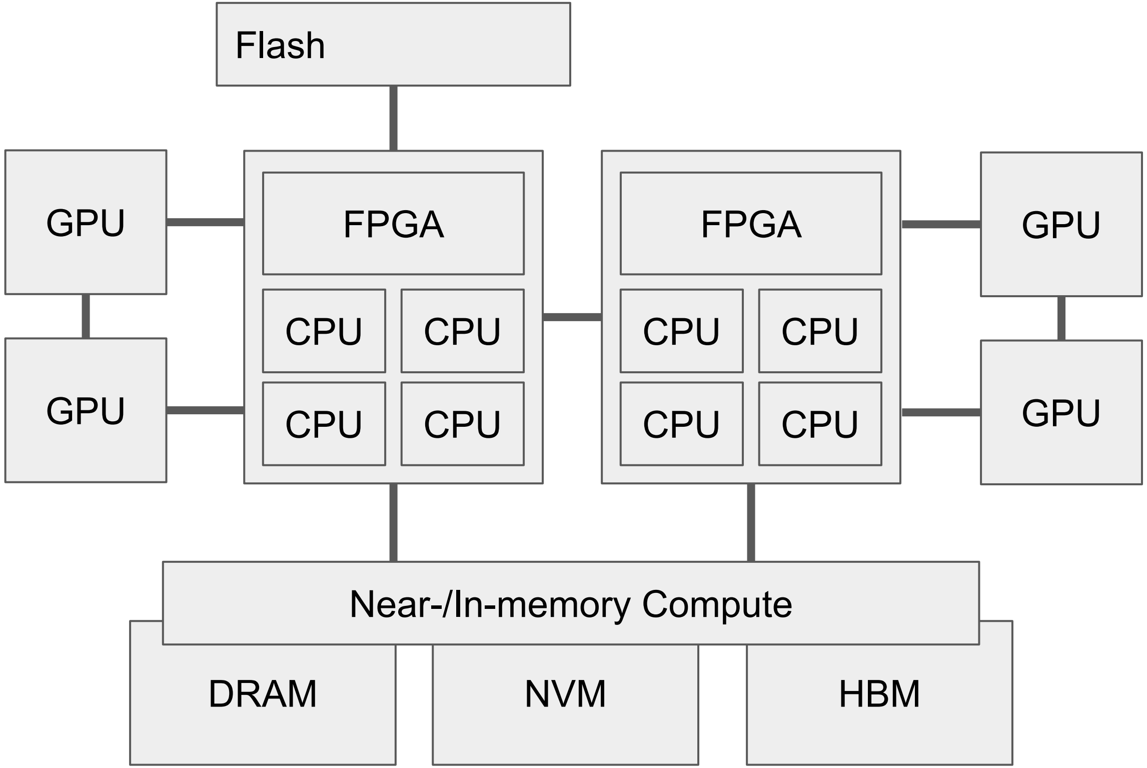
Out-of-Order execution
Vectorization
Cache effects
Multi-threading
Non-uniform memory access
Bulk-synchronous programming



Out-of-Order execution
Vectorization
Cache effects
Multi-threading
Non-uniform memory access
Bulk-synchronous programming
Communication
Data placement
Compute placement

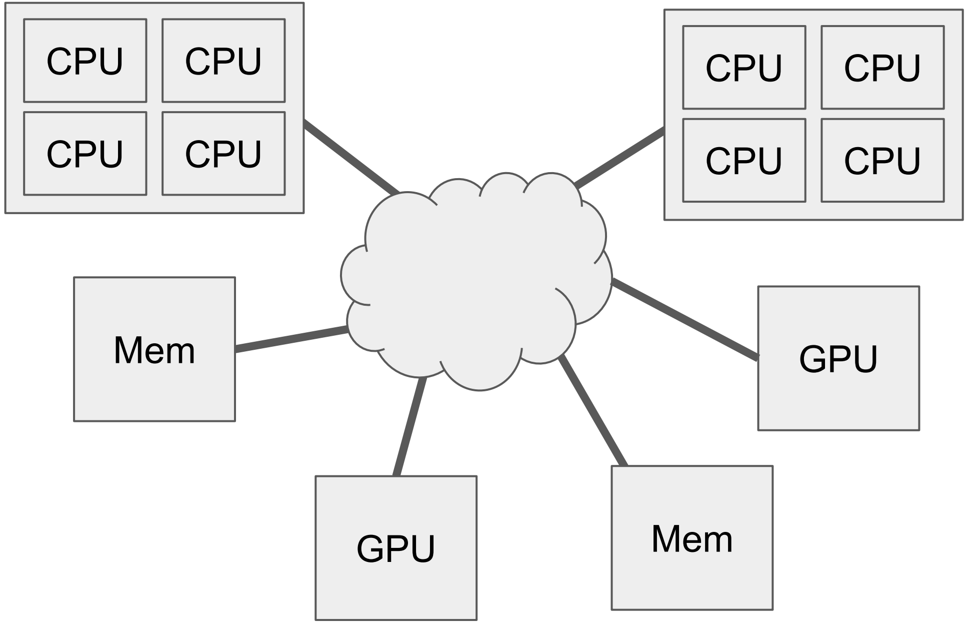


Out-of-Order execution
Vectorization
Cache effects
Multi-threading
Non-uniform memory access
Bulk-synchronous programming
Communication
Data placement
Compute placement
...
...
...



System Graph

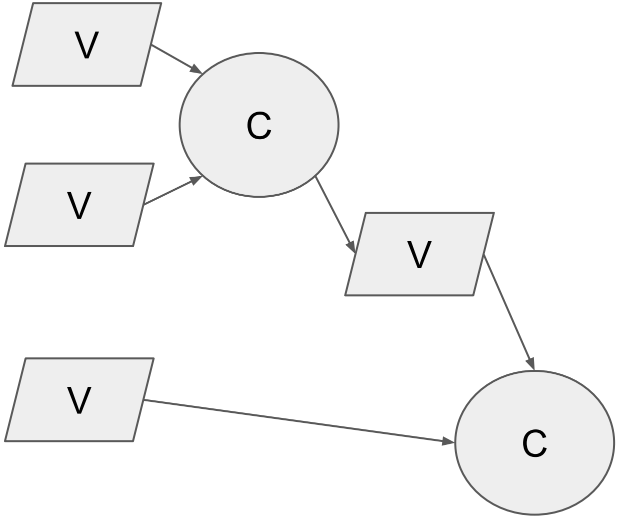
Communication
Data placement
Compute placement

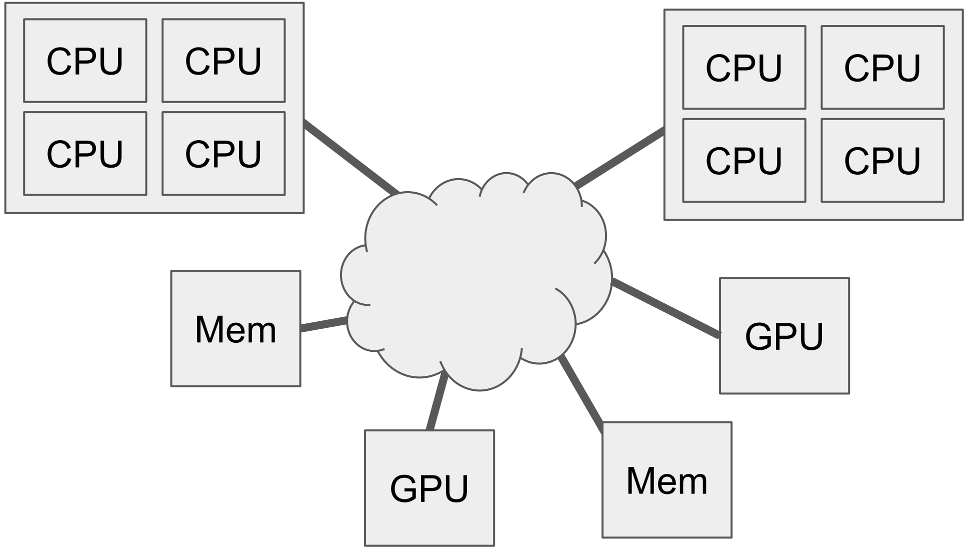


Graph:
  Nodes: {Compute, Storage}
  Edges: {Communication Links}

Application Graph

System Graph





Nodes: {Compute, Values}
Edges: {Dependence}

Values have a **size**

Nodes: {Compute, Storage}
Edges: {Communication Links}

Edges have a **performance model**

System Characterization

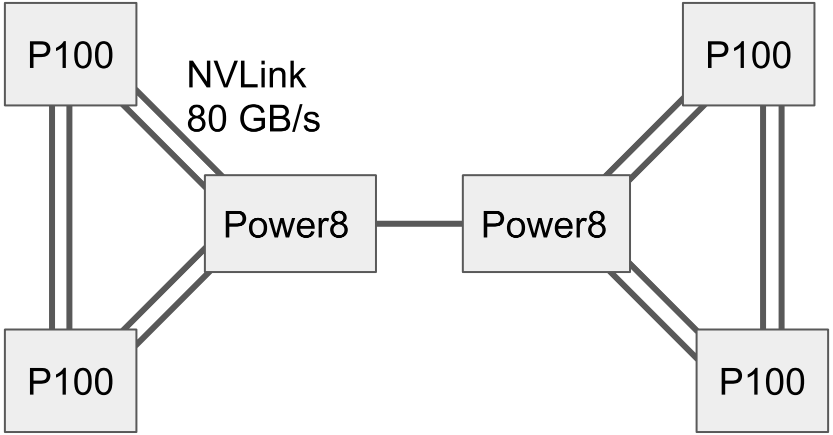
1) **Enumerate system topology**

Hwloc[1]

Portable interface for hardware affinity

NVIDIA Management Library

Cutting-edge NVIDIA hardware



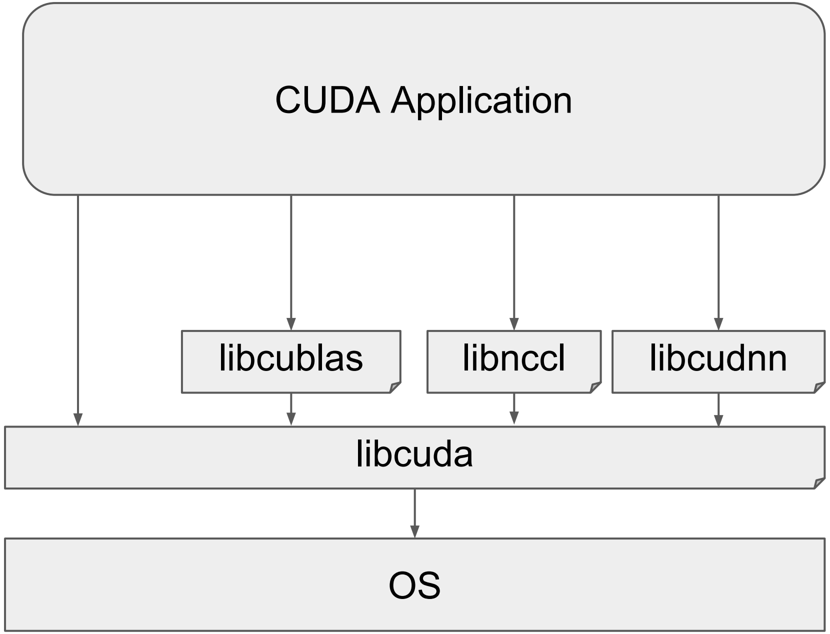
IBM "Minsky" Topology

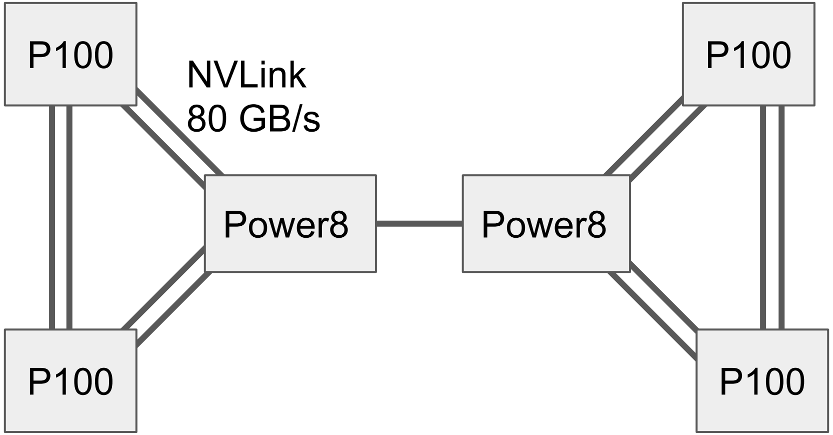
(not shown: network interfaces, disks, PCIe devices…)

[1] Broquedis, François, et al. "hwloc: A generic framework for managing hardware affinities in HPC applications." Parallel, Distributed and Network-Based Processing (PDP), 2010 18th Euromicro International Conference on. IEEE, 2010.

System Characterization

2) Characterize communication

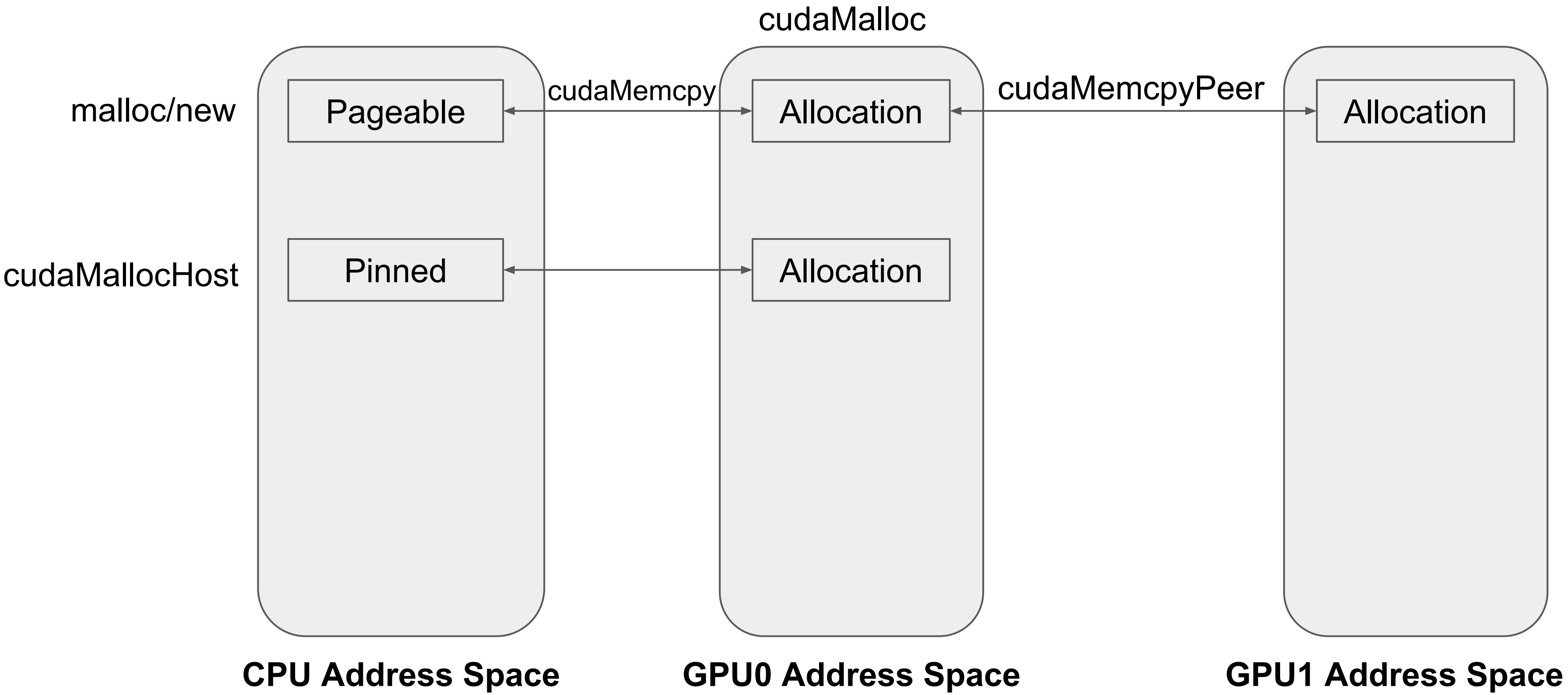




IBM “Minsky” Topology

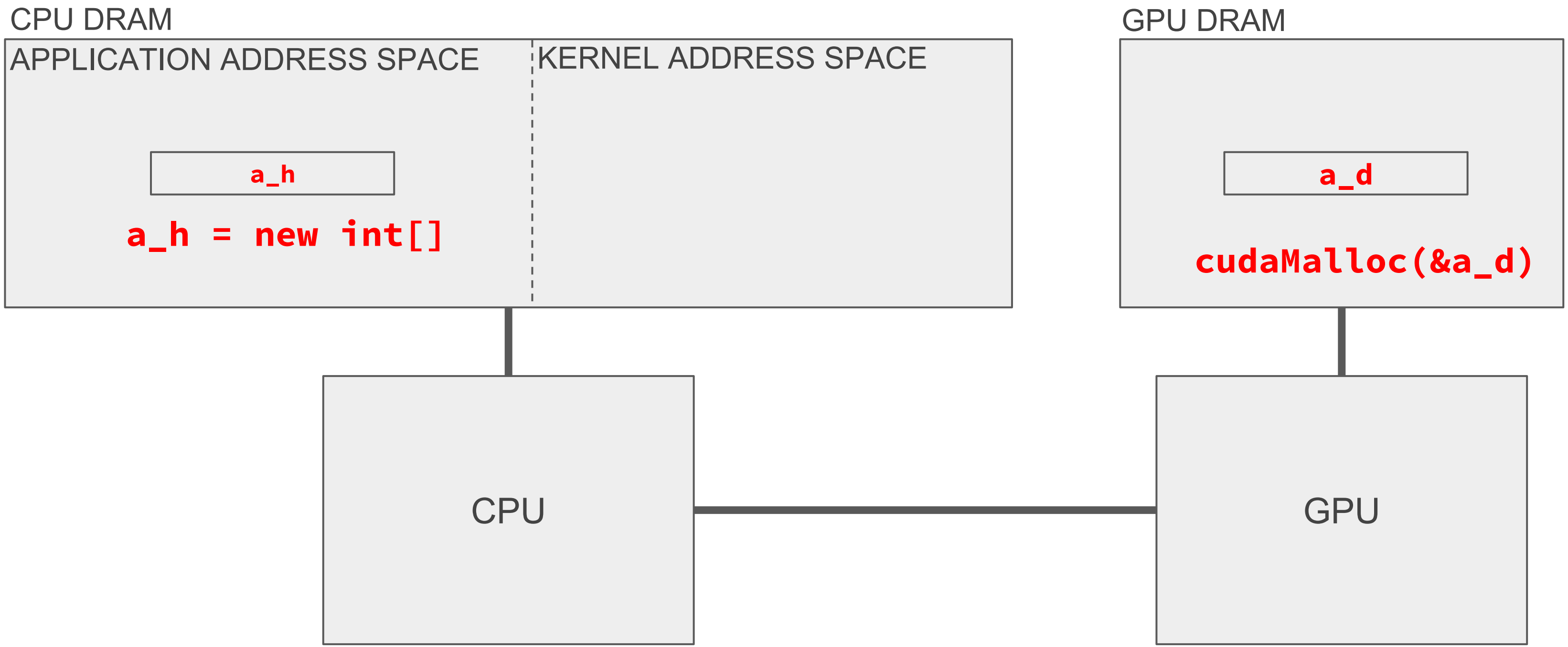
(not shown: network interfaces, disks, PCIe devices…)

CUDA 1-2 / CC 1.x :: Early CUDA



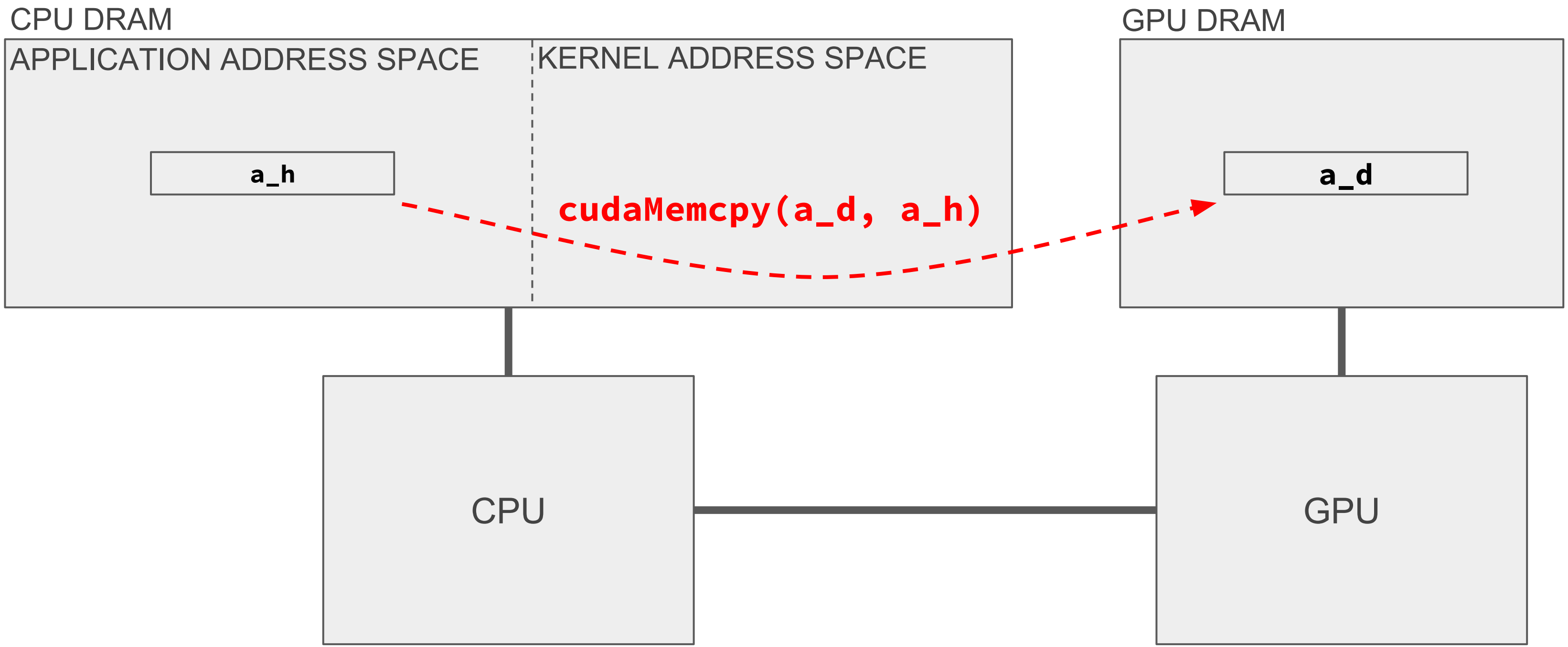
CUDA Memcopy Flow (Pageable)

1) Allocate pageable memory



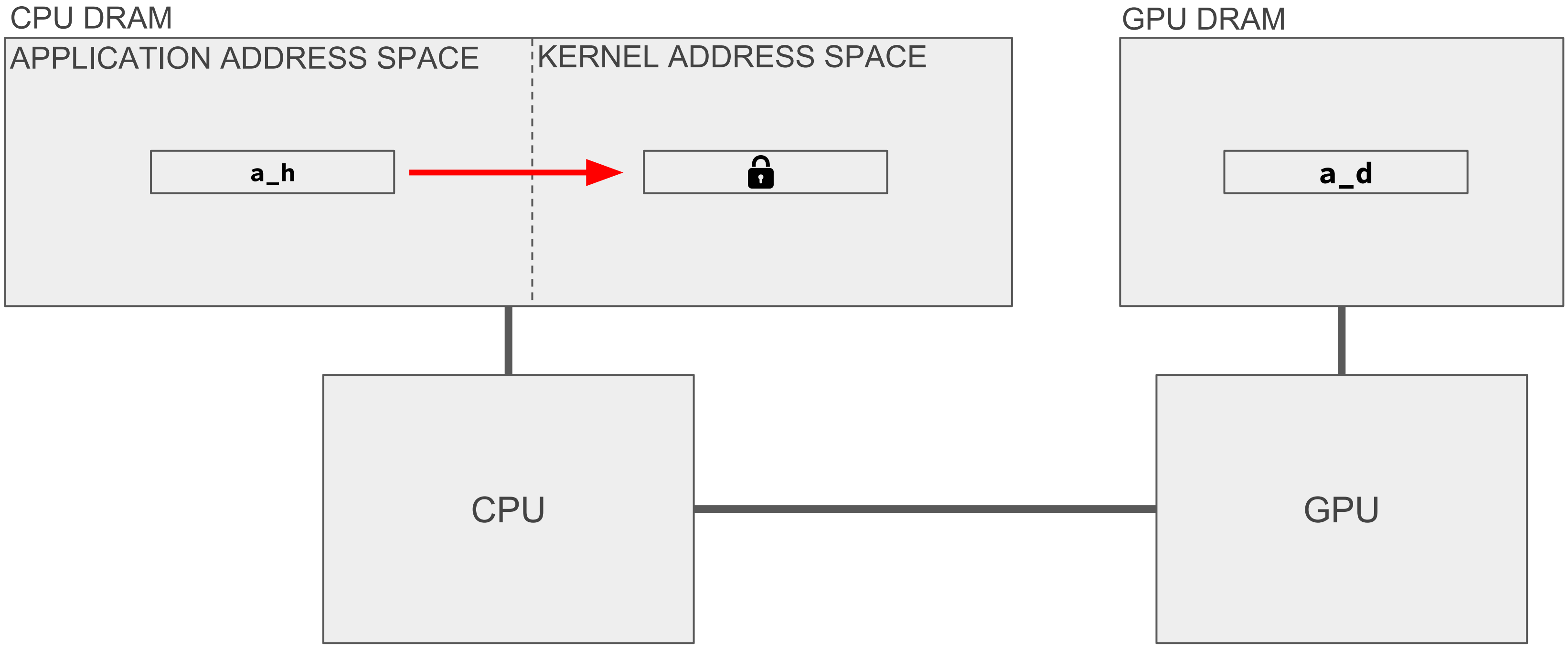
CUDA Memcopy Flow (Pageable)

2) Initiate CUDA Memcpy



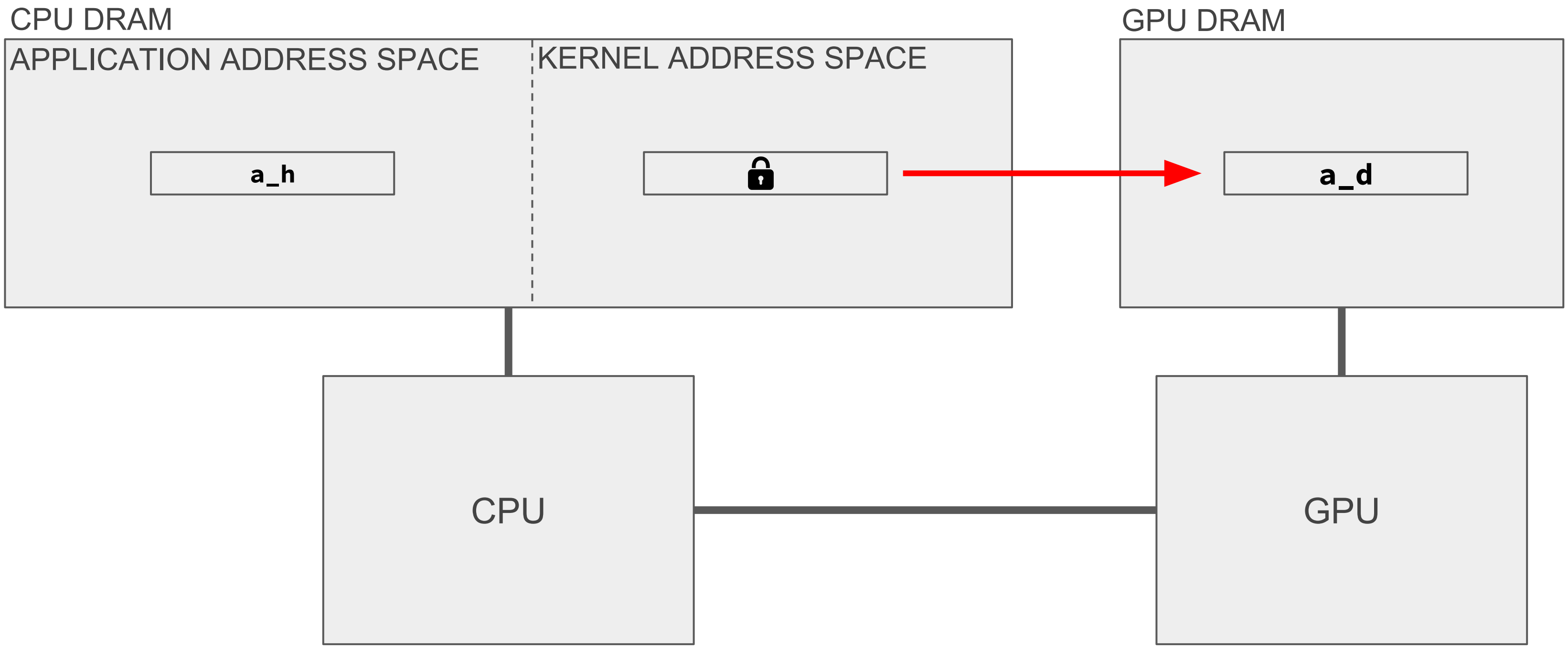
CUDA Memcopy Flow (Pageable)

3) Driver copies to pinned internal buffer



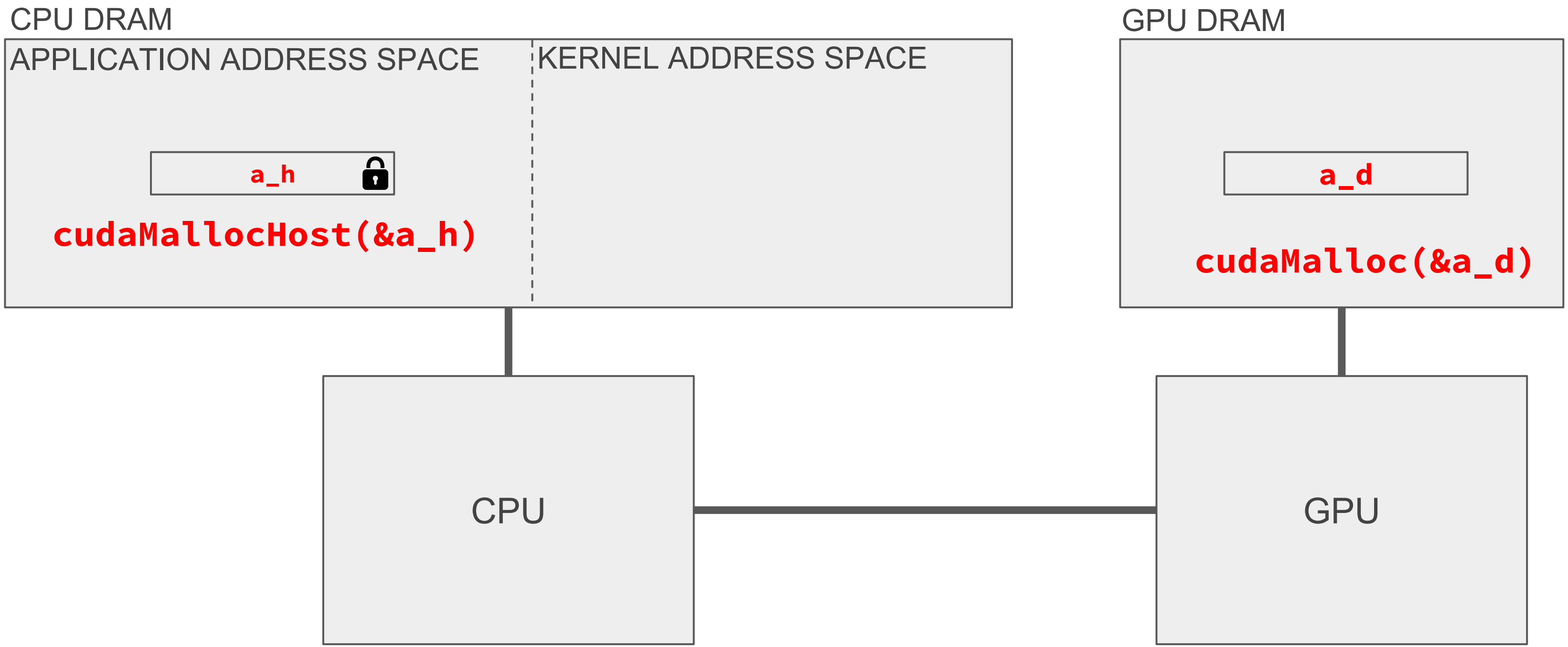
CUDA Memcopy Flow (Pageable)

4) CPU instructs GPU to begin **Direct Memory Access** copy



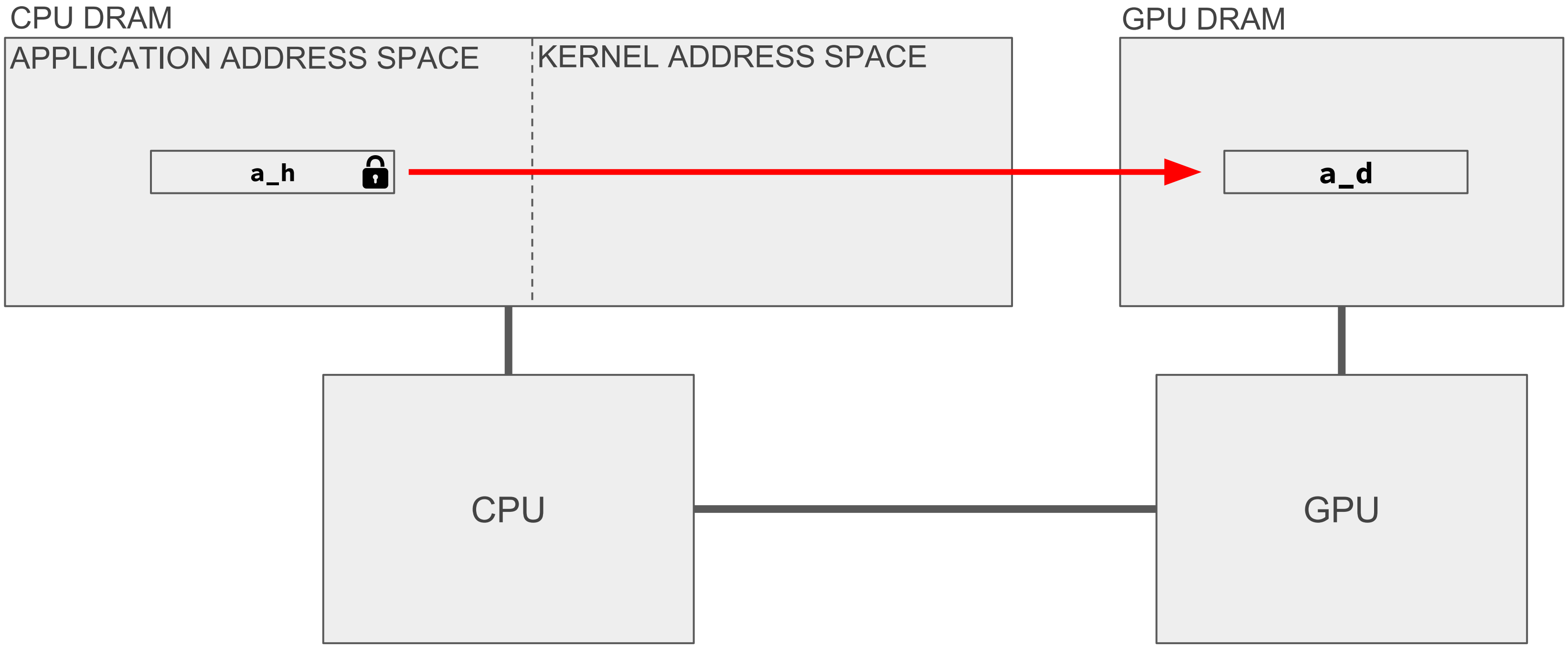
CUDA Memcopy Flow (Pinned)

1) Allocate pinned memory

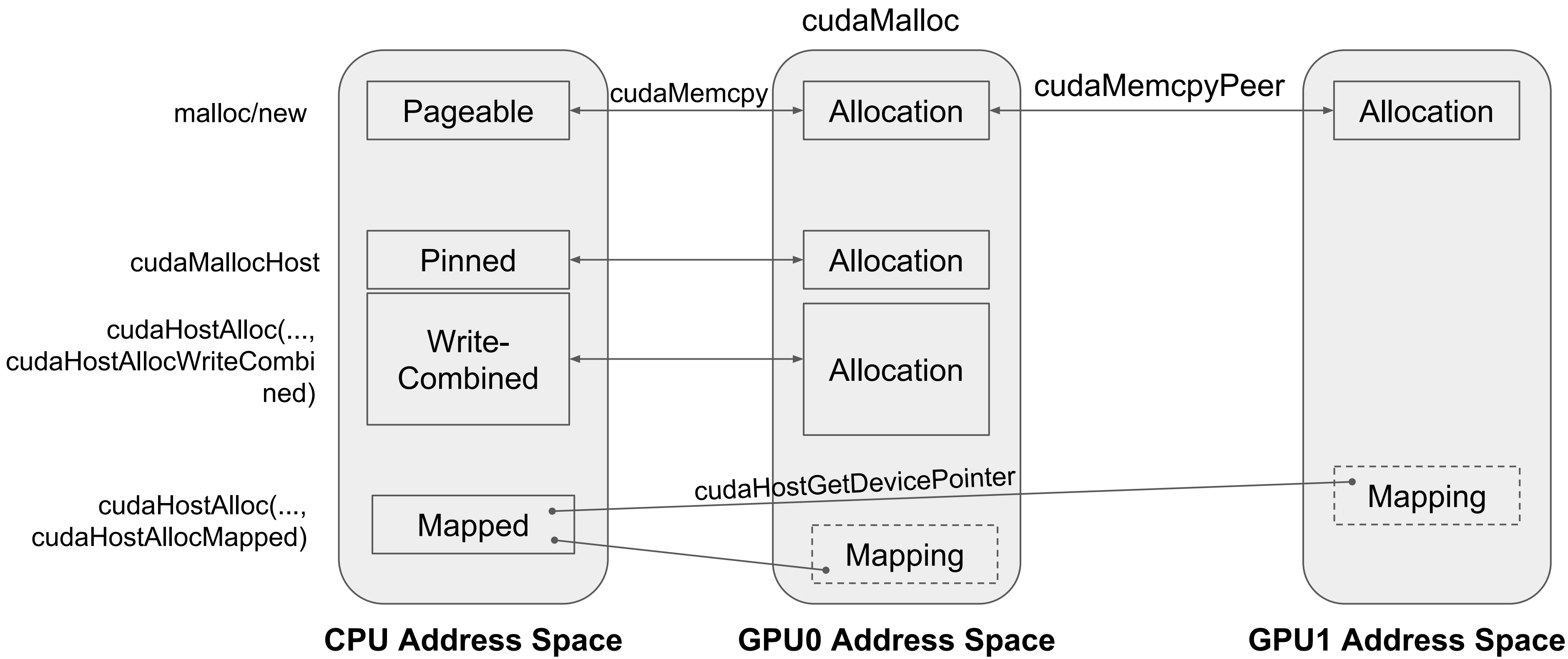


CUDA Memcopy Flow (Pinned)

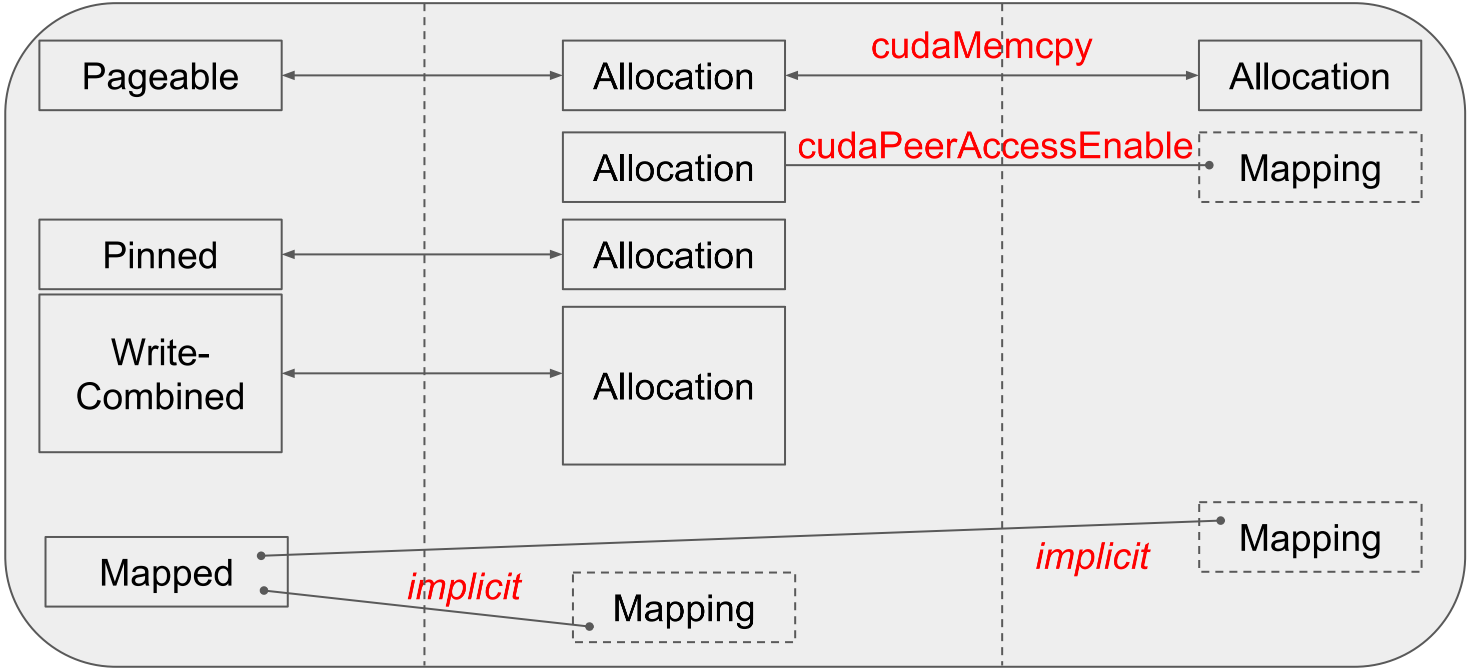
2) CPU instructs GPU to begin **D**irect **M**emory **A**ccess copy



CUDA 3 / CC 1.x :: Basic CUDA

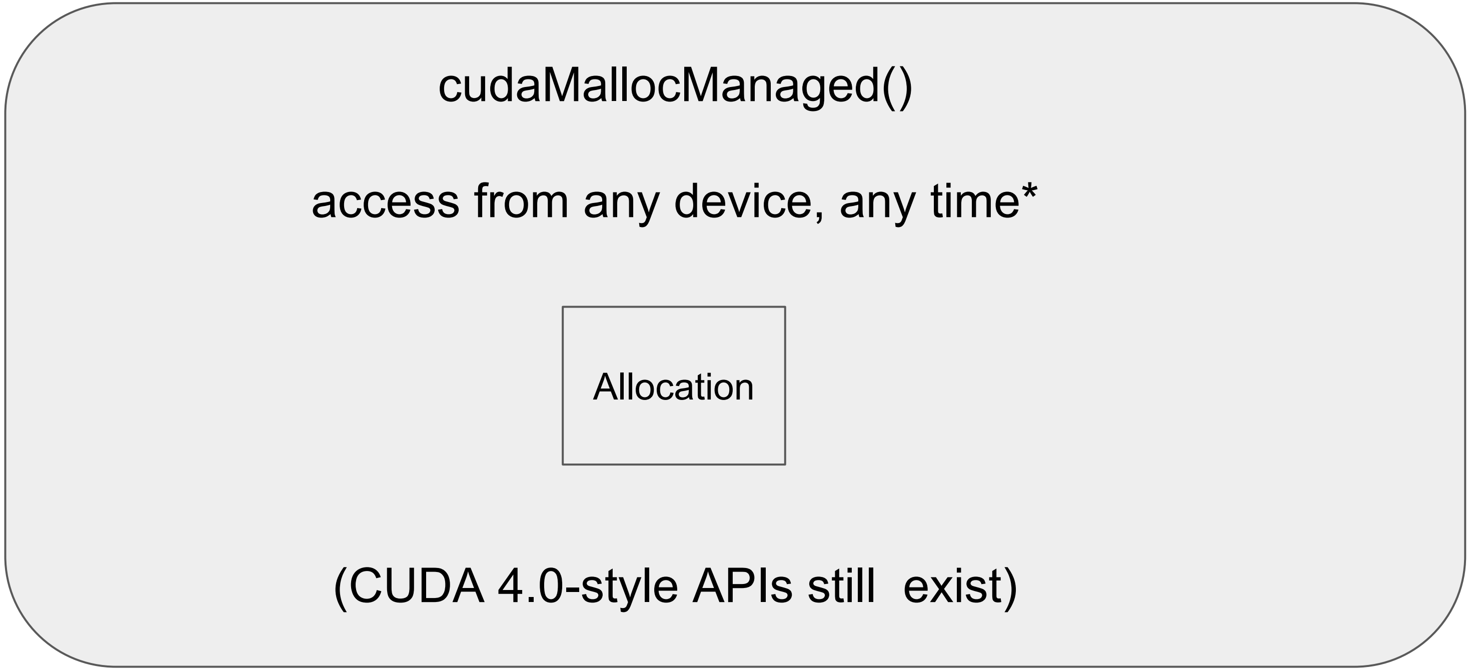


CUDA 4.0 / CC 2.0+ :: Unified Virtual Addressing



Unified Address Space

CUDA 6.0 / CC 3.0+ :: Unified Memory



cudaMallocManaged()

access from any device, any time*

Allocation

(CUDA 4.0-style APIs still exist)

* some restrictions apply, see CUDA Programing Guide for details

CUDA 6.0 / CC 3.0+ :: Unified Memory

Bulk transfers on access

	GPU 0	GPU 1	CPU

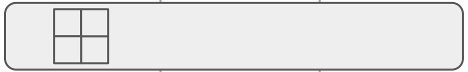
cudaSetDevice(0);
cudaMallocManaged(&a,...);
	a		

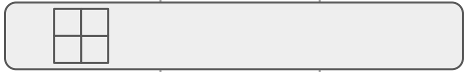
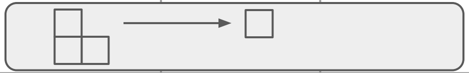
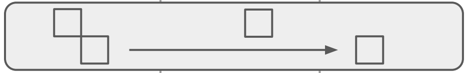
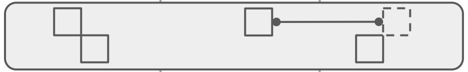
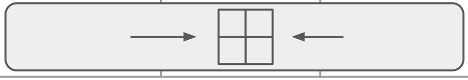
a[**page0**] = 0; // **gpu0**	a		
cudaDeviceSynchronize(); a[**page1**] = 1; // **gpu1**		→ a	
cudaDeviceSynchronize(); a[**page2**] = 2; // **cpu**			→ a

CUDA 8.0 / CC 6.0+ :: Unified Memory

Page faults on access
Automatic prefetching (not shown)

	GPU 0	**GPU 1**	**CPU**	

cudaSetDevice(0);
cudaMallocManaged(&a,...);
				

a[**page0**] = 0; // **gpu0**				
a[**page1**] = 1; // **gpu1**				Page fault and migration
a[**page2**] = 2; // **cpu**				Page fault and migration
cudaMemAdvise(a, **gpu1**, cudaMemAdviseSetPreferredLocation); a[**page1**] = 1; **// cpu**				Write served over NVLink
cudaMemPrefetcAsync(a, **gpu1**);				Bulk page migration

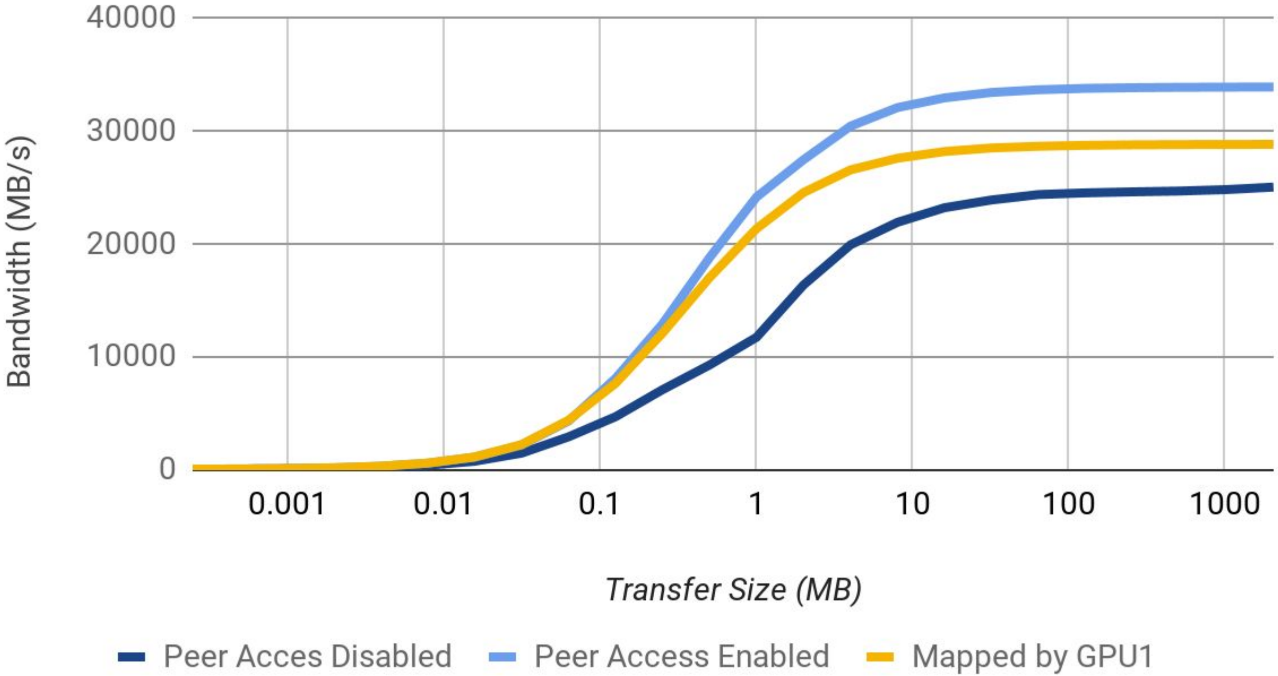
CUDA 9.2 / CC 7.0+ / Power9 / NVLink2 :: Unified Memory

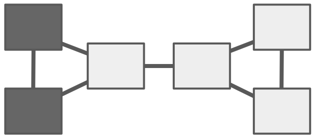
Address Translation Services: GPU can access CPU page tables

Access Counters for mapping vs migration

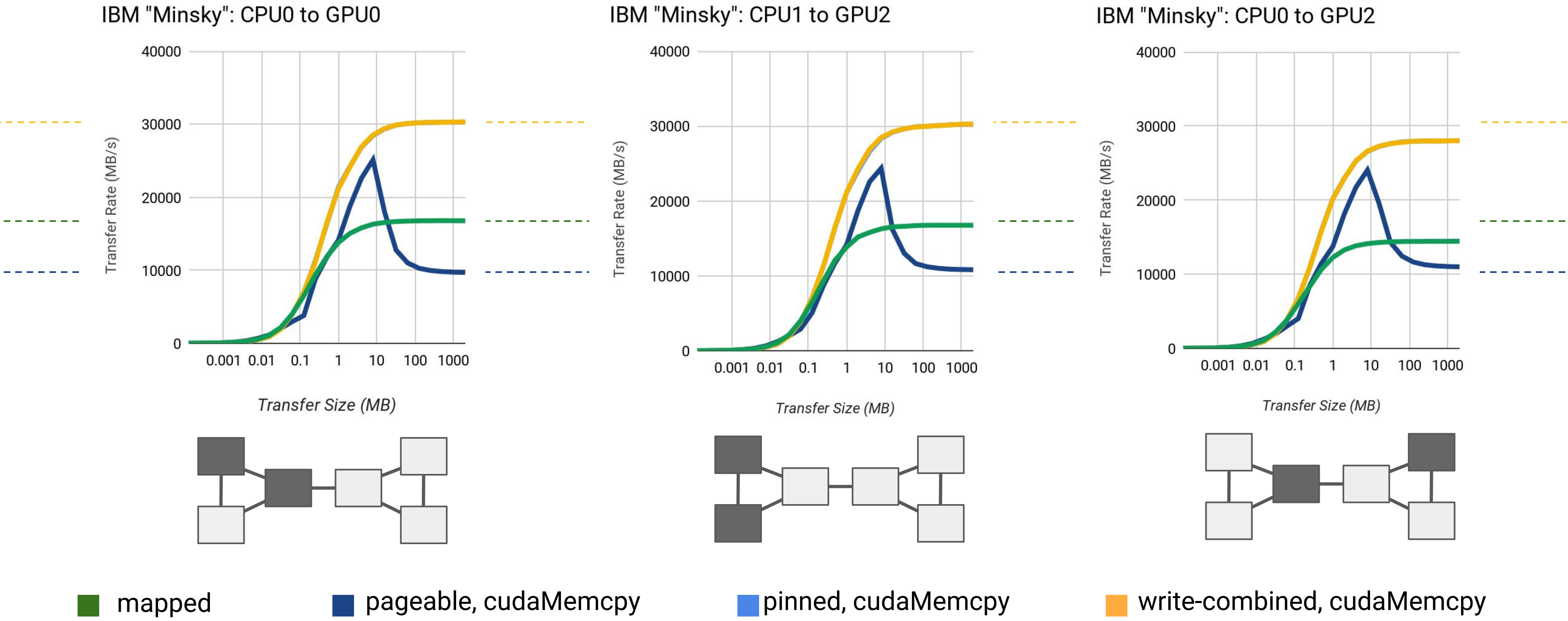
Atomic operations over NVLink2

IBM "Minsky" Transfer Rates, GPU0 to GPU1

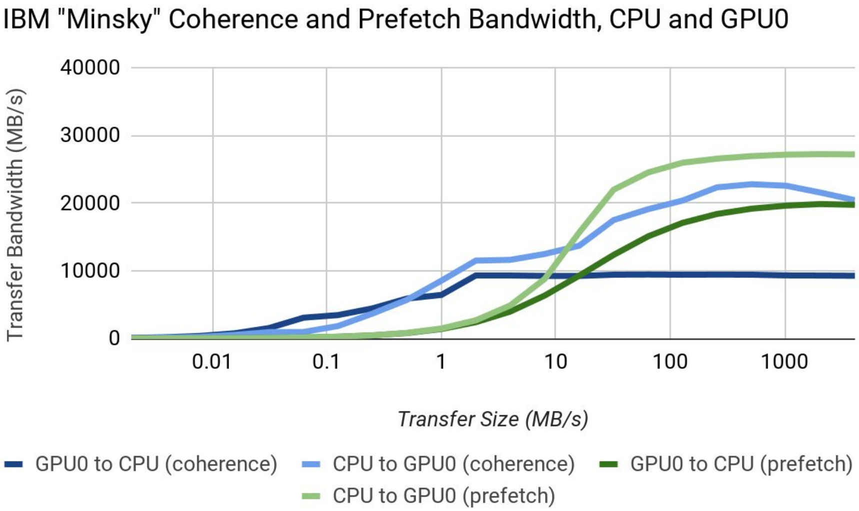


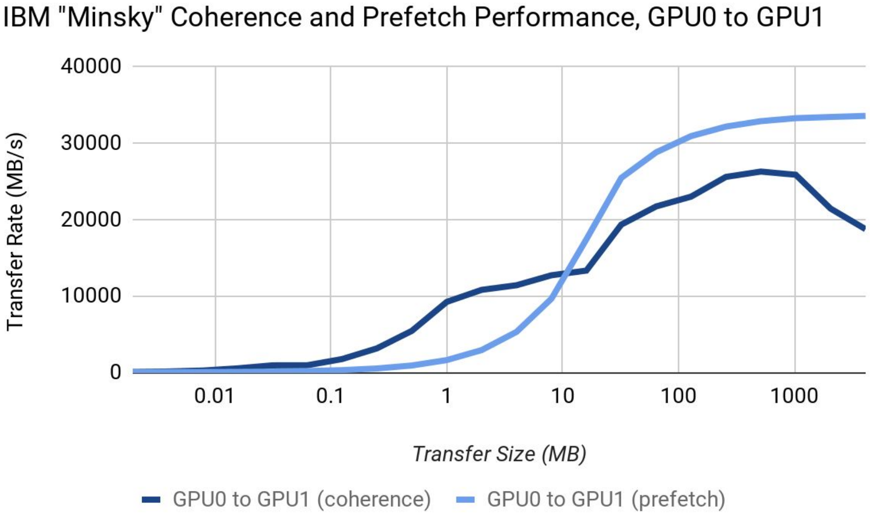


Identical CUDA Memcopies

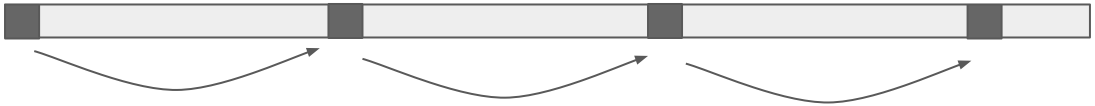


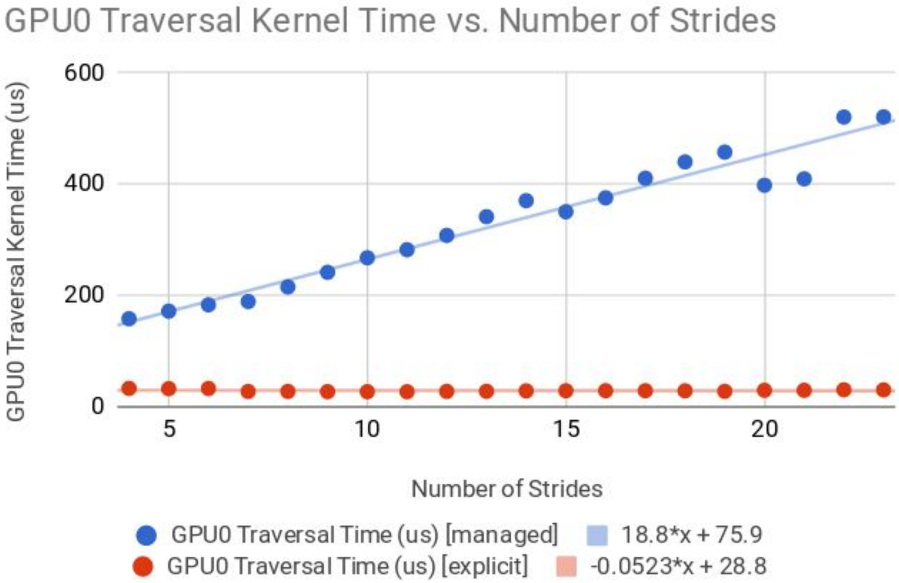
Prefetch Bandwidth and Demand Bandwidth





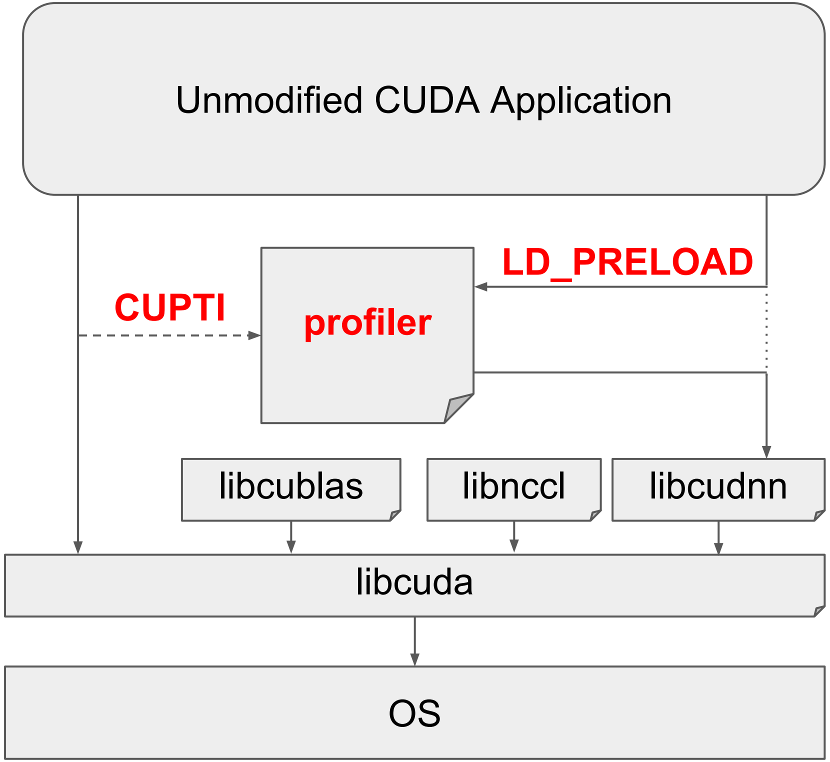
IMB “Minsky” Page Fault Latency (CPU to GPU0)




GPU0 Traversal Kernel Time vs. Number of Strides
GPU0 Traversal Kernel Time (us)
600
400
200
0
5
10
15
20
Number of Strides
● GPU0 Traversal Time (us) [managed]
18.8*x + 75.9
● GPU0 Traversal Time (us) [explicit]
-0.0523*x + 28.8

~20µs per page fault

Software Characterization



- LD_PRELOAD
  - Observe generic shared library calls
- **CU**DA **P**rofiling **T**ools **I**nterface
  - Observe CUDA runtime calls

```
 1: {"build":"20180328-223829+0000","git":"dirty","version":"0.1.0"}
 2: {"device":0,"name":"cudaSetDevice",...
 3: {"name":"cudaMalloc","ptr":140667466547200,"size":409600,...
 4: {"name":"cudaMalloc","ptr":140667466956800,"size":819200,...
 5: {"name":"cudaMalloc","ptr":140667467776000,"size":819200,...
 6: {"count":409600,"dst":140667466547200,"name":"cudaMemcpy","src":140668104151056,"...
 7: {"count":819200,"dst":140667466956800,"name":"cudaMemcpy","src":140667584897040,"...
 8: {"block_dim":...,"grid_dim":...,"name":"cudaConfigureCall","shared_mem":0,"stream":0,...
 9: {"arg":140667467776000,"name":"cudaSetupArgument","offset":0,"size":8,...
10: {"arg":140667466547200,"name":"cudaSetupArgument","offset":8,"size":8,...
11: {"arg":140667466956800,"name":"cudaSetupArgument","offset":16,"size":8,...
12: {"arg":12855032555119837504,"name":"cudaSetupArgument","offset":24,"size":4,...
13: {"arg":1374389535360,"name":"cudaSetupArgument","offset":28,"size":4,...
14: {"func":4216851,"name":"cudaLaunch"}
```

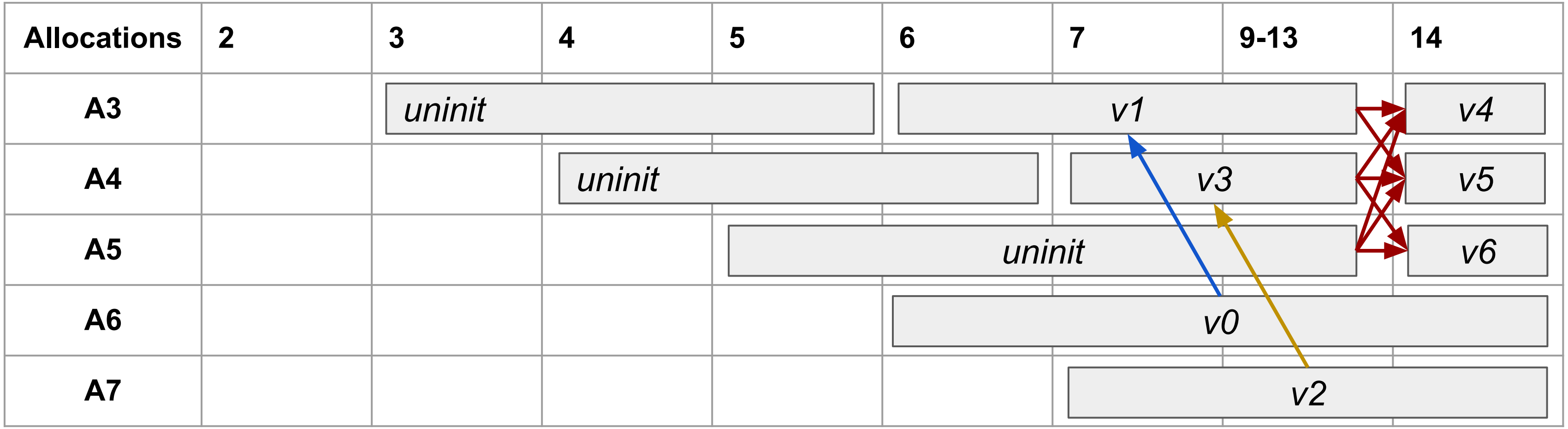
PROFILE FORMAT

ENGLISH

```
2: device 0 is associated with calls from this thread
3: Allocation A3 @ 140667466547200, 409600 bytes
4: Allocation A4 @ 140667466956800, 819200 bytes
5: Allocation A5 @ 140667467776000, 819200 bytes
6: Infer >= 409600 allocation A6 @ 140668104151056
    Copy A6 -> A3
7: Infer >= 819200 allocation A6 @ 140668104151056
    Copy A7 -> A4
9-13: Record arguments to upcoming kernel (including A3, A4, A5)
14: Launch kernel @ address 4216851
```

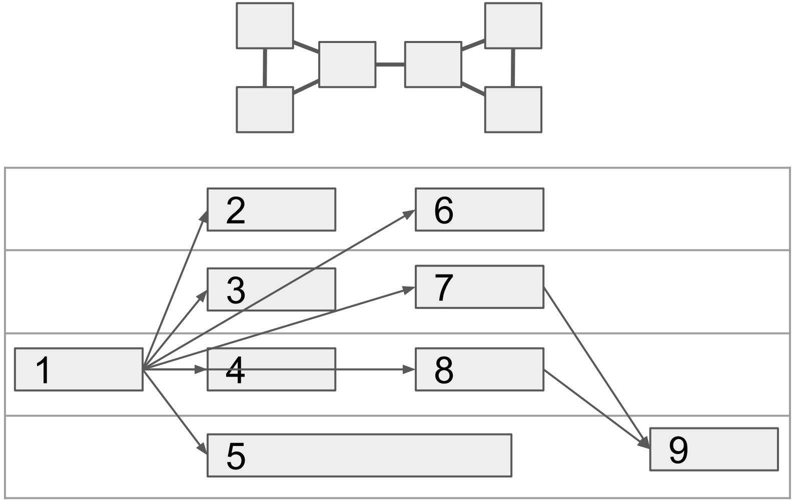
Dynamic Dependence Graph from Profile

```
3: Allocation A3 @ 140667466547200, 409600 bytes
4: Allocation A4 @ 140667466956800, 819200 bytes
5: Allocation A5 @ 140667467776000, 819200 bytes
6: Infer >= 409600 initialized allocation A6 @ 140668104151056
   Copy A6 -> A3
7: Infer >= 819200 initialized allocation A6 @ 140668104151056
   Copy A7 -> A4
9-13: Set up arguments (including A3,A4,A5)
14: Launch function 4216851
```

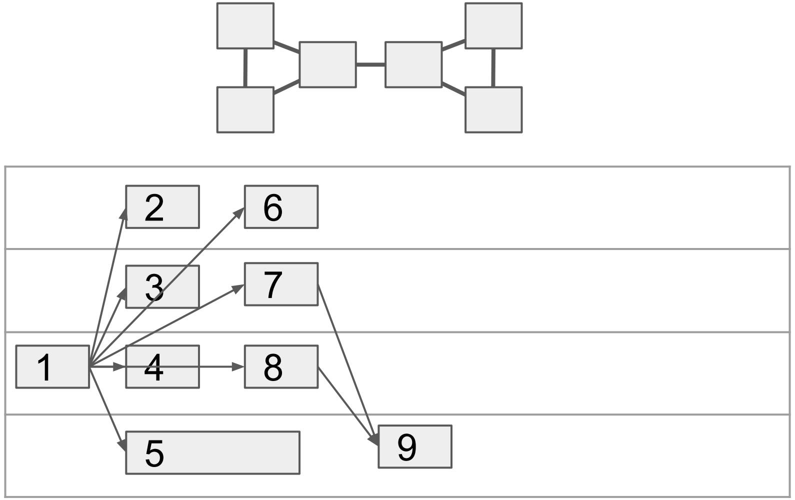


Replay with Tweaked Components

IBM “Minsky” w/ Power8, P100s, NVLink



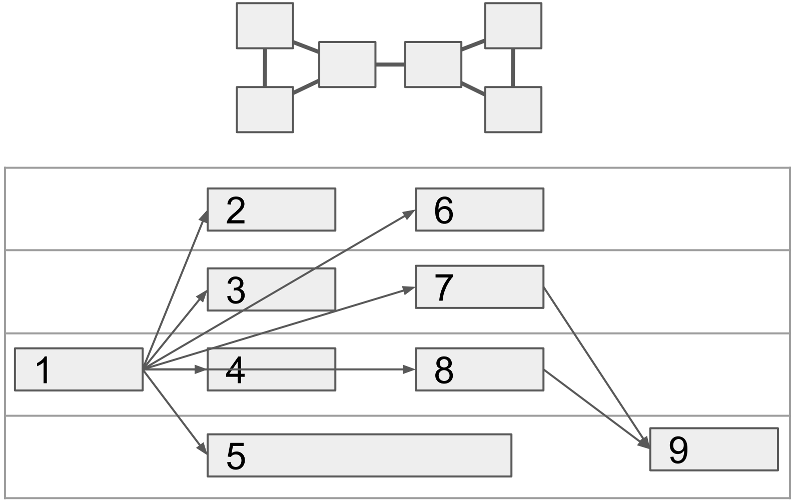
IBM w/ Power9, V100s, NVLink2



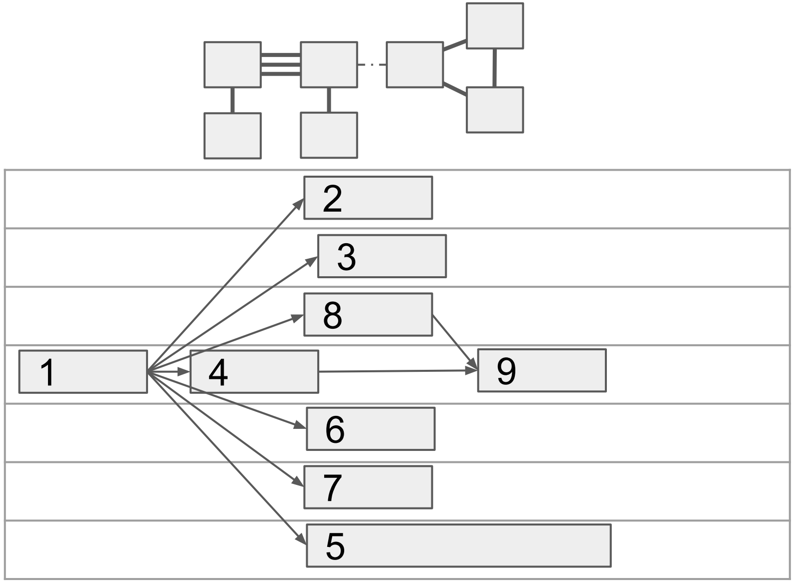
Kernel performance model?

Replay with Tweaked Components

IBM “Minsky”
Power8, P100s, NVLink



Illinois *FutureArch*
RISCV, V100, NuLink, FPGA, NVM



Scheduling problem
Heterogeneous performance model

Questions / Comments / Concerns

Special thanks to the Center for Cognitive Computing Systems Research (C3SR), NCSA, and Dominic Grande.