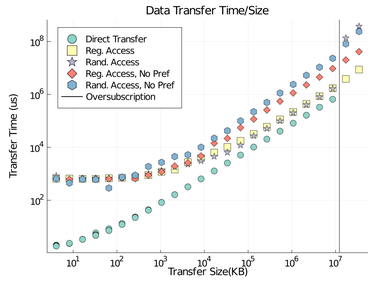
Holistic Performance Analysis and Optimization of Unified Virtual Memory

Tyler Allen, Rong Ge (Adviser)

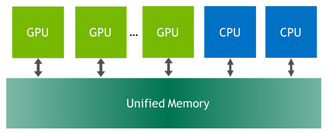
Problem and Motivation

- **Problem:** UVM improves programmer productivity at the cost of performance. Can we have both?
  - Big Picture: UVM is increasingly popular - used in HPC applications and frameworks such as Kokkos, Raja, and Trilinos.
  - SotA: UVM performance loss not well studied at **systems software, system architecture** level.
- **Our Approach:** Holistic Analysis, then Optimization
  - Study full system – systems software, workload generation, applications.
  - Include advanced features – prefetching and oversubscription - for real-world use-cases.
  - Apply optimizations at *method* level, applicable to other implementations besides NVIDIA UVM.



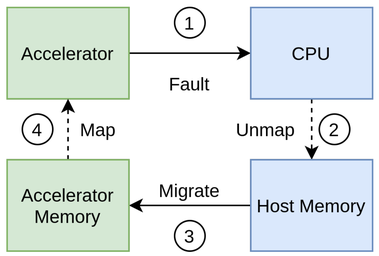
Background

- UVM is an example of *heterogeneous shared memory.*
- Heterogeneous shared memory systems fundamentally use *on-demand page migration* as the backing technology for physical memory placement.
  - Same programming view of memory on all devices.
  - UVM uses software *density prefetching b*ecause on-demand migration is prohibitively slow.
  - Enables device memory *oversubscription,* expanding capacity to host memory – similar to Swap storage.



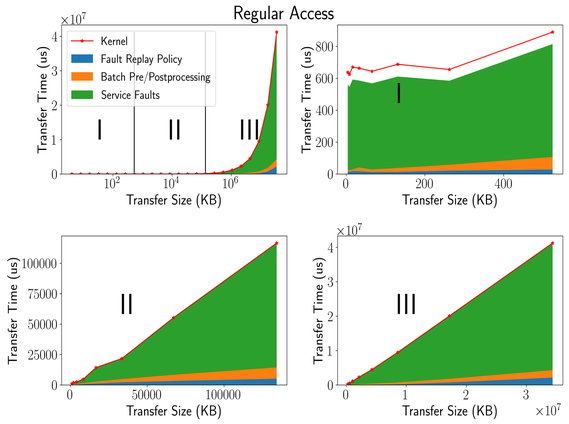
https://developer-blogs.nvidia.com/wp-content/uploads/2017/11/Unified-Memory-MultiGPU-FI.png

- All heterogeneous shared memory systems are fundamentally 4-steps: ***fault→unmap→migrate→map***



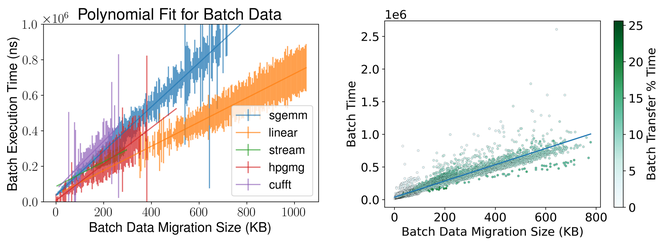
High-Level Performance

- The majority of cost is the "service faults" category.
  - "Service faults" is primarily the **unmap → migrate → map** process – we focus further study here.



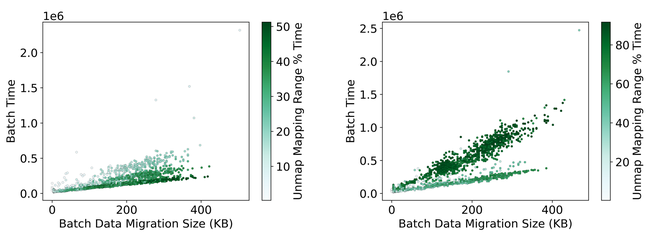
Impact of Data Migration

- **Key Finding:** Most of UVM cost is *software* overhead.
- The amount of data migrated is highly correlated with the time required to process, but (left):
  - Applications unexpectedly show different trends for migration costs.
  - Fault batches within the same application have varied costs.
- Data transfer is not the majority of cost (right).
  - **Interconnect hardware bandwidth is not the main problem**.



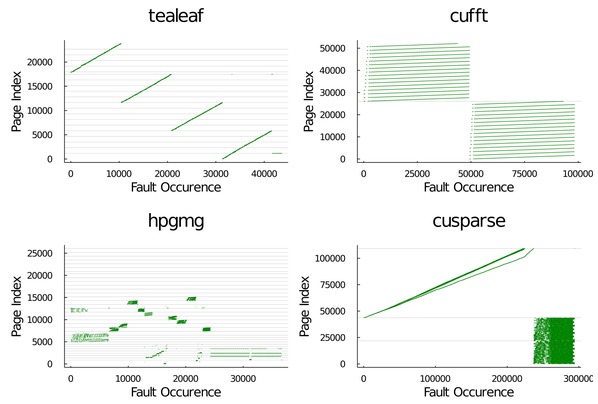
Page Unmapping

- **Key Finding:** A primary source of overhead is CPU page unmapping.
- This overhead can account for up to 50% of batch execution time (left), and over 80% in certain scenarios with CPU parallelization (right).
- Tracing indicates that this overhead is likely attributed to TLB shootdowns.
  - Optimizing TLB shootdowns for migration has broad impacts.



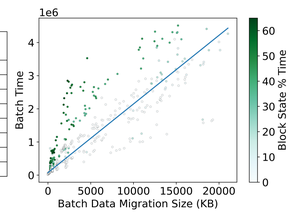
Prefetching

- Page-level access pattern of several applications show the workload of this prefetcher:



- Prefetching reduces the total page faults by up to 96% (left).
- **Key Finding:** Prefetching is very effective but cannot remove overhead for lower-level system functionality.
  - Costs like **page unmapping** and DMA setup (right) **cannot be avoided through the existing prefetcher** and are a larger percentage of overhead with prefetching.
- Prefetching notably decreases the number of *batches* seen, reducing the baseline overhead despite bulk costs.
- The trade-offs for prefetching increase when oversubscription is enabled, potentially increasing the amount of evicted data.

	total faults	faults w/ prefetching	fault reduction (%)
Regular	2493569	442011	82.27
Random	2522931	51558	97.95
sgemm	6522314	223998	96.56
stream	3721584	578884	84.44
cufft	101494	10074	90.07
tealeaf	1193619	394148	66.97
hpgmg	139785	50231	64.06
cusparse	2342572	611719	73.88



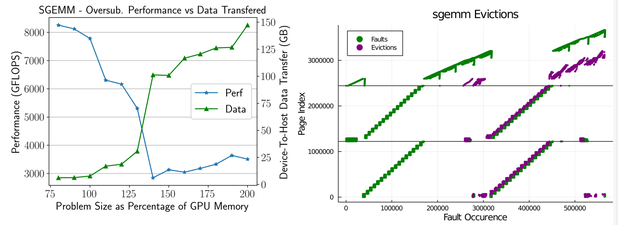
Optimization

- Optimization focusing on page unmapping and system-level improvements is in-progress and to be added in September.
- Optimizations focus on TLB shootdowns specifically for data migration, impacting heterogeneous shared memory systems but also other migration scenarios, such as swap storage.

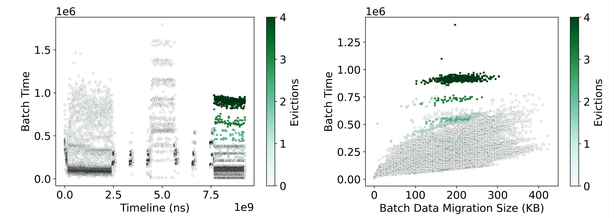
Placeholder for Migration TLB Shootdown Optimization Results	Placeholder for Migration TLB Shootdown Optimization Results

Oversubscription

- Eviction uses LRU from host driver perspective.
  - Page fault raises virtual address block (2MB) in LRU list.
- Highly-used regions can be evicted early.
  - Fully-resident regions cannot move up in LRU list.
- Evicted pages must be paged back (left), and quantity of evicted data is highly correlated with performance (right).



- Eviction occurs once memory is fully allocated. (left)
- **Key Finding:** Eviction has flat cost based on number of virtual address blocks evicted (right).



Conclusions/Discussion

- Data transfer is not the primary cost of UVM; higher bandwidth can improve performance but is not the main issue.
- System costs like building DMA connections and TLB shootdowns are a primary source of overhead.
- Prefetching is effective but cannot eliminate these overheads.
- Oversubscription/eviction has flat cost and does not antagonize system costs, but is subject to them.
- Provided unmapping optimizations improve baseline application performance without source code changes.
- Results applicable to other related systems, such as Heterogeneous Memory Management (HMM).

Publications

- Tyler Allen, Rong Ge, "In-Depth Analyses of Unified Virtual Memory System for GPU Accelerated Computing", Supercomputing 2021
- Tyler Allen, Rong Ge, "Demystifying GPU UVM Cost with Deep Runtime and Workload Analysis," 2021 IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2021, pp. 141-150, 10.1109/IPDPS49936.2021.00023

Student Author | This Poster | Reproducibility