Memory-Aware Scheduling Of Tasks Sharing Data On Multiple GPUs
Maxime Gonthier, Loris Marchal, Samuel Thibault

To cite this version:
Maxime Gonthier, Loris Marchal, Samuel Thibault. Memory-Aware Scheduling Of Tasks Sharing Data On Multiple GPUs. ISC 2023 - ISC High Performance 2023, May 2023, Hamburg, Germany. Lecture Notes in Computer Science. hal-04090595
Memory-Aware Scheduling of Tasks Sharing Data On Multiple GPUs

Maxime GONTHIER - Samuel THIBAULT - Loris MARCHAL - maxime.gonthier@ens-lyon.fr - samuel.thibault@inria.fr - loris.marchal@ens-lyon.fr

Motivation: Extract peak performance from GPUs

- Scientific application uses a huge quantity of data
- GPUs can calculate quickly but their memory are limited
- The PCI bus becomes a bottleneck when large amounts of data need to be loaded

Framework and objectives

- Independent tasks sharing data
- Same size data
- Multiple GPUs with limited memory

Algorithms from the state of the art

- Using (hyper-)graph partitioning: hMETIS
  - Hypergraph = Represents a data being shared by multiple tasks
  - hMETIS produces subsets of task that maximize the amount of edges (data shares) inside each subset
  - We add dynamic reordering and task stealing to improve performance
- EAGER
  - Process tasks in the order in which it receives them = our baseline
- Deque Model Data Aware Ready (DMDAR):
  - Schedules tasks where their completion times are expected to be minimal. Uses a ready strategy to favor tasks whose data has already been loaded into memory.

A novel strategy: Data-Aware Reactive Task Scheduling (DARTS)

Experimental evaluations on independent tasksset

- Tesla V100 GPUs with a 12000 MB/s PCI bandwidth
- GPUs can calculate quickly but their memory is 500 MB
- 10 runs per experiment (deviation < 2%)

Experimental evaluations on independent taskset

- DMDAR suffers from a conflict between prefetches and evictions
- DARTS+LUF stays close to the asymptotic goal

Experimental evaluations with dependencies

- DARTS+LUF stays close to the asymptotic goal
- DMDAR suffers from a conflict between prefetches and evictions
- DARTS can be used with a reduced complexity (DARTS+OPTI) by stopping the search for the optimal data earlier.

Summary and conclusion

- High-performance computing applications, like weather and climate forecasting, have an increasing demand in computer power
- GPUs have a large computation power but have a limited memory. This, coupled with the share of a single PCI express bus by multiple GPUs induce a huge amount of data transfers which reduces performance
- Limiting data movement is crucial to extract the most out of GPUs

- We provide a new algorithm (DARTS) and a new eviction policy (LUF) focused on data locality
- DARTS+LUF achieves good performance because it:
  - Limits data transfers thanks to the finding of an optimal data and an adapted eviction policy
  - Overlaps communication and computations by distributing transfers over time
  - Can be used with a reduced complexity
  - Can deal with priorities
- Areas for improvement:
  - Manage multiple MPI nodes
  - Use other out-of-core applications (DARTS is not specific to GPUs)

Motivations:

- Overlaps communication and computations by distributing transfers over time
- Our eviction policy: Least Used in the Future (LUF)
  - If possible, evict data not useful for any task in taskBuffer and used by a minimal number of tasks in plannedTasks
  - Otherwise, evict data whose next usage is the furthest in the future in plannedTasks
  - Update plannedTasks

Experimental evaluations with dependencies

- DARTS achieves better performances once the memory is a constraint (on the right of the green dotted line) and behaves as DMDAR when the memory is not limited (on the left). It also greatly reduces data transfers. DARTS outperforms the scheduler from PaRSEC, a state-of-the-art runtime.

Areas for improvement:

- Manage multiple MPI nodes
- Use other out-of-core applications (DARTS is not specific to GPUs)