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

HAL Id: hal-04090595
https://inria.hal.science/hal-04090595
Submitted on 5 May 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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

To get peak performance we need to minimize the number of access to the main memory

Framework and objectives

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

Objective 1: Minimize the number of tasks on each GPU
Objective 2: Minimize the amount of data loads on each GPU

Algorithms from the state of the art

- Using (hyper-)graph partitioning: hMETIS
  - Hypergraph \(\rightarrow\) 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 \(\Rightarrow\) 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)

- Task flow
  1. DARTS fills Planned Tasks
  2. Some tasks are popped by Task Buffer and start to be prefetched
  3. If the GPU is idling, a task is popped from Task Buffer
  4. After an eviction, tasks in Planned Tasks can be removed

- Strategy
  1. Find the optimal data \(D\) such that the number of tasks depending on \(D\) and on other data in memory is maximum
  2. Fill plannedTasks with these tasks

- 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 on independent tasksset

- Tesla V100 GPUs with a 12000 MBs PCI bandwidth
- GPU memory limited to 500 MB to better distinguish the performance even on small datasets
- 10 runs per experiment (deviation + 2%)

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

Experimental evaluations with dependencies

- Pathological matrix size due to LRU’s eviction policy when the memory is a constraint (after the red dotted line) for EAGER, hMETIS and DARTS
- DMDAR suffers from a large scheduling cost
- DARTS can be used with a reduced complexity (DARTS+OPTI) by stopping the search for the optimal data earlier.
- Even without memory limitation, DARTS outperforms DMDAR
- hMETIS suffers from a large scheduling cost
- DARTS+OPTI can accommodate both input matrices
- Orange dotted line: maximum number of transfers that can be done during the minimum time for computation \(\rightarrow\) the hard limitation induced by the PCI bus bandwidth

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 large amount of data transfers which reduces performance \(\Rightarrow\) 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 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.

- Results: 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/)

- We need to optimize the data locality of the tasks to reduce the number of data loads on each GPU.