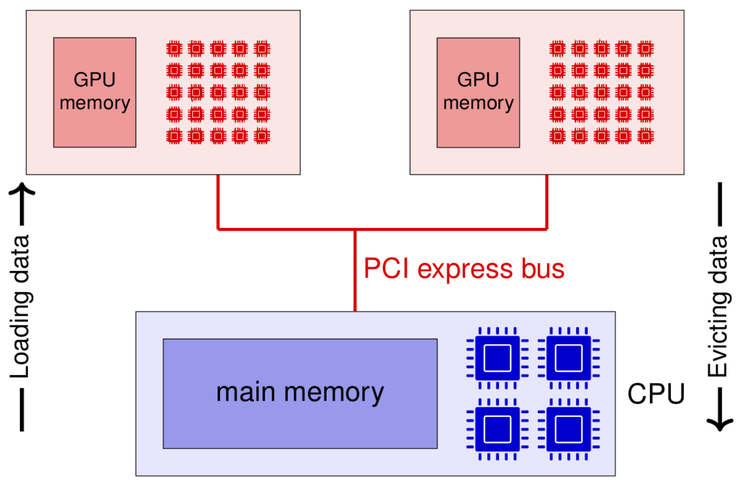
Memory-Aware Scheduling Of Tasks Sharing Data On Multiple GPUs

Maxime Gonthier, Samuel Thibault, Loris Marchal

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

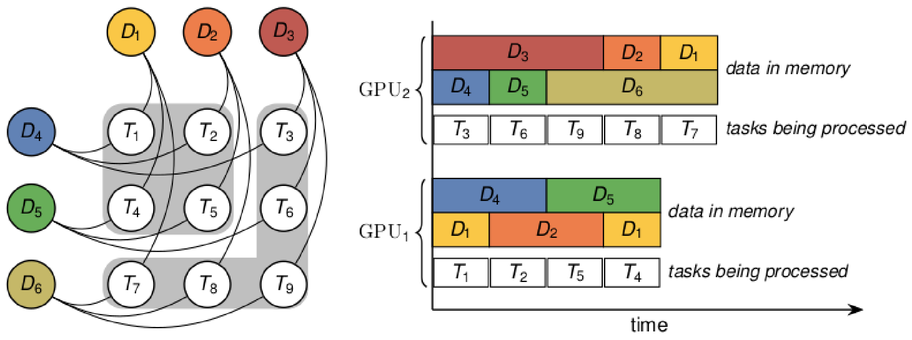
Motivation: Extract peak performance from GPUs



- 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

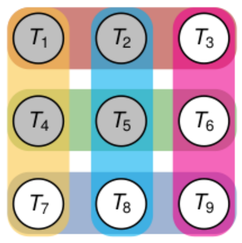


- Independant tasks sharing data
- Same size data
- Multiple GPUs with limited memory

Figure: Example with a memory of size 2 data. The graph of input data dependencies is shown on the left. The figure on the right corresponds to the partition and schedule produced by the scheduler

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 ⇒ 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

Figure: A hyper-graph with 2 partitions (grey and white)

- **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)

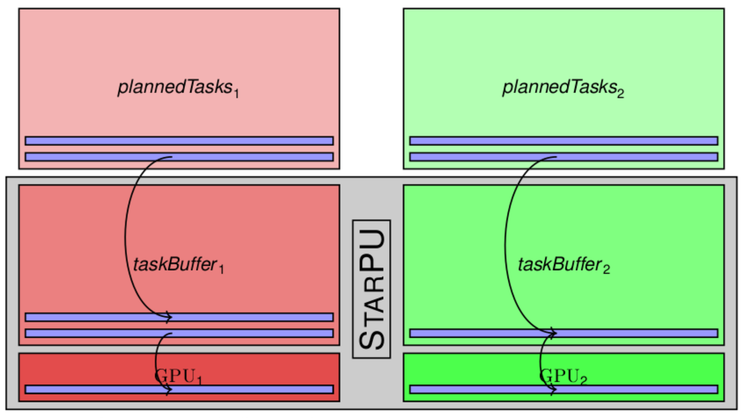


Figure: Task flow when using DARTS

- **2 queues of tasks**
  - Tasks in plannedTasks can be removed.
  - Tasks in taskBuffer are being prefetched and can't be removed
  - Tasks in GPU are being computed
- **Flow of tasks**
  - (1). DARTS fills plannedTasks
  - (2). Some tasks are popped by taskBuffer and start to be prefetched
  - (3). If the GPU is idling, a task is popped from taskBuffer

Intuition : Consider data locality before task allocation

- **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, apply Belady's rule on tasks already allocated
  - Update plannedTasks

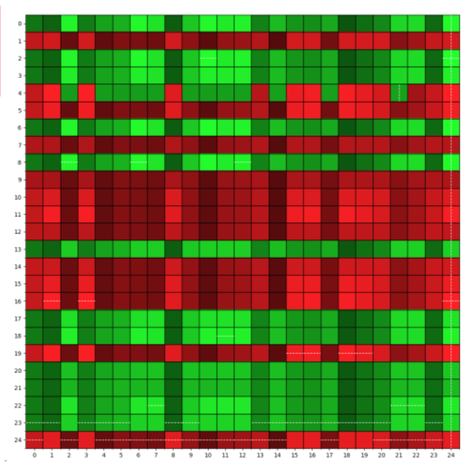


Figure: Visualization of DARTS processing order on the tiled Matrix Multiplication

Experimental settings

- Tesla V100 GPUs
- 12000 MB/s PCI bandwidth
- GPU memory limited to 500 MB to better distinguish the performance of different strategies even on small datasets

Experimental evaluations

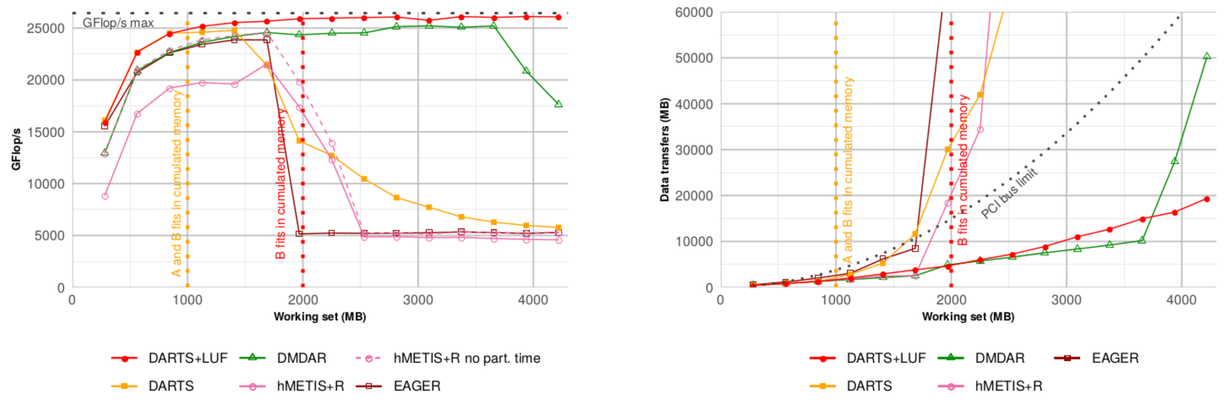


Figure: Gflop/s (left) and data transfers (right) on a tiled (1 x n) Matrix Multiplication in outer product order with 2 Tesla V100 GPUs

The dotted horizontal black line represents the maximum Gflop/s the GPUs can achieve when processing elementary matrix product and is our asymptotic goal.
The red dotted vertical line denotes the situation when the cumulated GPUs memory can fit exactly only one of the two input matrices, and the orange line denotes the situation when it can accommodate both input matrices.

The black diagonal dotted line on the figure on the right is the maximum number of transfers that can be done during the minimum time for computation, thus the hard limitation induced by the PCI bus bandwidth.

- 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 conflict between prefetches and evictions
- DARTS+LUF stay close to the asymptotic goal !

DARTS nicely overlaps communications and computations

- Large scheduling cost for all algorithms
- DARTS+OPTI is a variant that reduces the complexity by stopping the search for the optimal data earlier

DARTS+OPTI is able to get good performance with a reduced scheduling quality (and complexity)

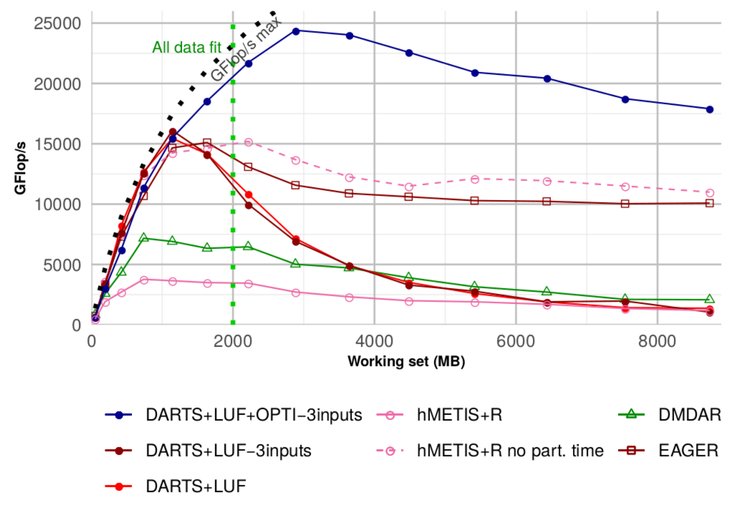
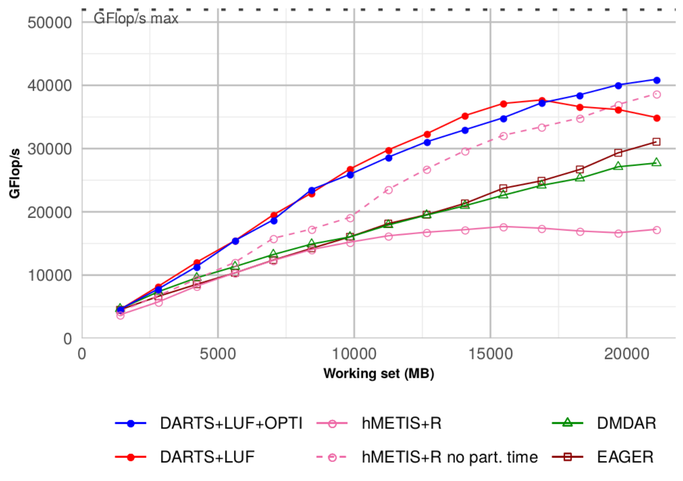


Figure: Cholesky decomposition (without dependancies) with 4 Tesla V100 GPUs



- hMETIS suffers from a large scheduling cost

DARTS produces a processing order that best distributes transfers over time
⇒ Better performance even without memory limitation

Figure: Tiled sparse Matrix Multiplication in outer product order without memory limitation (32GB by GPU) with 4 Tesla V100 GPUs

Summary and conclusion

- **Motivations :**
  - 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 multilple GPUs induce a large 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
- **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
- **Areas for improvement:**
  - Consider tasks with dependencies
  - Consider inter-GPU communications (NVLink)
  - Manage multiple MPI nodes

This work is available in the proceedings of the 36th IEEE International Parallel & Distributed Processing Symposium