GPU-Tasking à la Carte?

Eventify Meets GPUs

Laura Morgenstern [1,2], Ivo Kabadshow[1],
Matthias Werner[2]
[1] Forschungszentrum Jülich
[2] Chemnitz University of Technology

Appetizers

1. Motivation

- **Task-parallelism** omnipresent in data mining, machine learning, matrix factorization and molecular dynamics
- **Final goal:** uniform task-parallel programming technology for strong scaling on heterogeneous systems
- **Idea:** extend task-parallel programming library Eventify [1] to GPUs

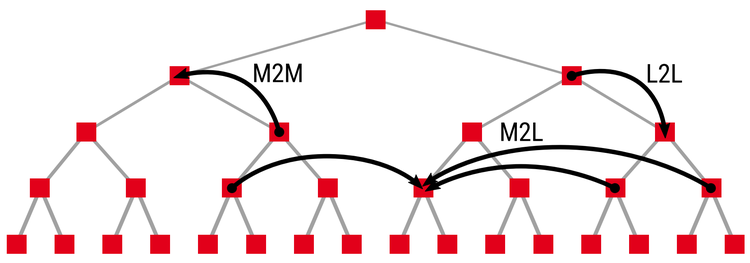
2. Requirements

- **Scalability:** Enable fine-grained task parallelism to exploit the full parallelization potential of the algorithm.
- **Portability:** Preferably uniform code path for CPU and GPU code from the algorithm developer's point of view.

3. Use Case

Our use case is a Fast Multipole Method for molecular dynamics which consists of the following task types:

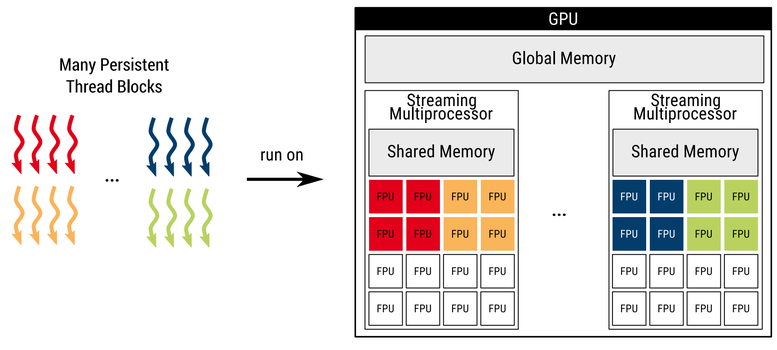
1. **P2M:** Expand particles on lowest level into multipole moments
2. **M2M:** Transfer multipole moments upwards
3. **M2L:** Translate multipole moments into local moments
4. **L2L:** Transfer local moments downwards
5. **L2P:** Translate local moments to particles on lowest level
6. **P2P:** Evaluate near field interactions



Main Courses

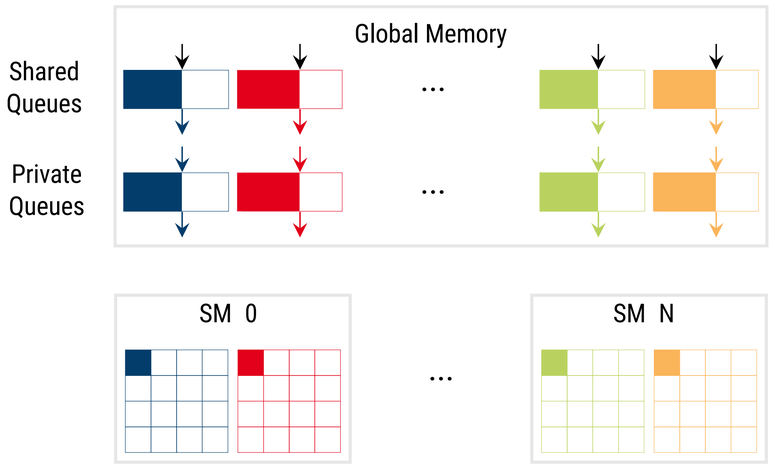
4. Uniform Programming Model

- **Goal:** bridge gap between CPU and GPU programming
- **Technical concept:**
  - Employ a persistent thread [2] kernel for each task type T (currently, T ∈ {M2M, M2L, L2L})
  - Each persistent thread kernel produces and consumes tasks of its template type T
  - Task execution workflow is the same for each task type
  - Only one kernel launch per task type



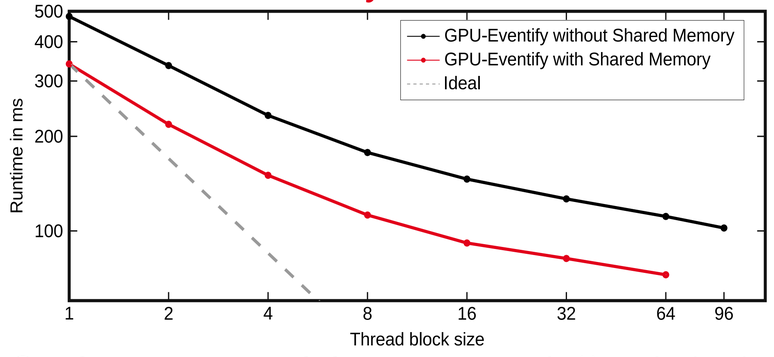
5. Multiple Hierarchical Task Queues

- Each persistent thread block owns two queues:
  - A **shared queue** that enables work sharing with other thread blocks
  - A **private queue** that enables lock-less queue access for all threads of the owning thread block



Desserts

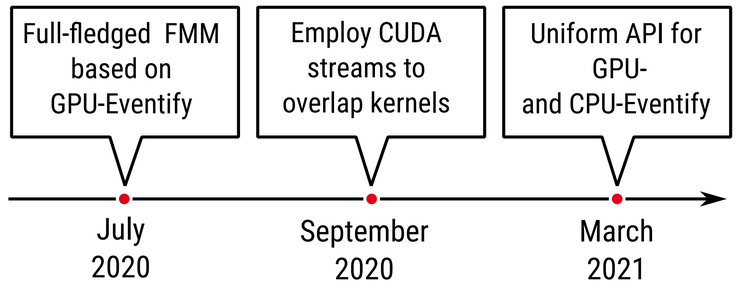
6. Performance Analysis



- **Software:** GPU-Eventify for a Fast Monopole (!) Method with multipole order p=0 and tree depth d=6
- **Hardware:** NVIDIA Tesla V100 with 80 SMs
- Experimentally determined runtime-optimal grid size:
  - Without shared memory grid size g=320
  - With shared memory grid size g=480
- Best runtimes reached for largest thread block sizes
- Usage of shared memory improves runtime by 30%

Cherry On Top

7. Future Work



8. References

[1] D. Haensel, L. Morgenstern, A. Beckmann, I. Kabadshow and H. Dachsel. Eventify: Event-Based Task Parallelism for Strong Scaling. Accepted for publication at PASC20, 2020

[2] K. Gupta, J. A. Stuart and J. D. Owens. A study of Persistent Threads style GPU programming for GPGPU workloads. 2012 Innovative Parallel Computing (InPar), San Jose, CA, 2012, pp. 1-14.

TECHNISCHE UNIVERSITÄT
CHEMNITZ

JÜLICH
Forschungszentrum