

Contributions

Developing applications able to exploit the dazzling performance of GPUs (Graphics Processing Units) is not a trivial task, and becomes even harder when they have irregular data access patterns or control flows. Several approaches have been proposed to simplify GPU programming such as OpenMP, OpenACC. However they have a performance gap with native programming models as their compiler does not have comprehensive knowledge about how to transform code and what to optimize. This thesis targets directive-based programming models to enhance their capabilities for GPU programming.

My contributions are in three directions:

- Developed a Task Based Programming model (OmpSs+OpenMP), along with the compiler and runtime.
- Code transformation of nested parallelism for irregular applications such as sparse matrix operations, graph and graphics algorithms
- Compiler optimization for loop scheduling in GPU.

Thesis Software Contribution:

- Mercurium compiler, Clang Frontend , NVIDIA's PGI Compiler

Task Based GPU Offload Model (OmpSs+OpenMP)

MACC = Mercurium ACCelerator model

Introduces a new dialect programming model

Asynchronous Task based GPU Offload model

Combination OmpSs + OpenMP

Incorporate advantages of models

Ease to use accelerators

Its aim is to let compiler offload and parallelize

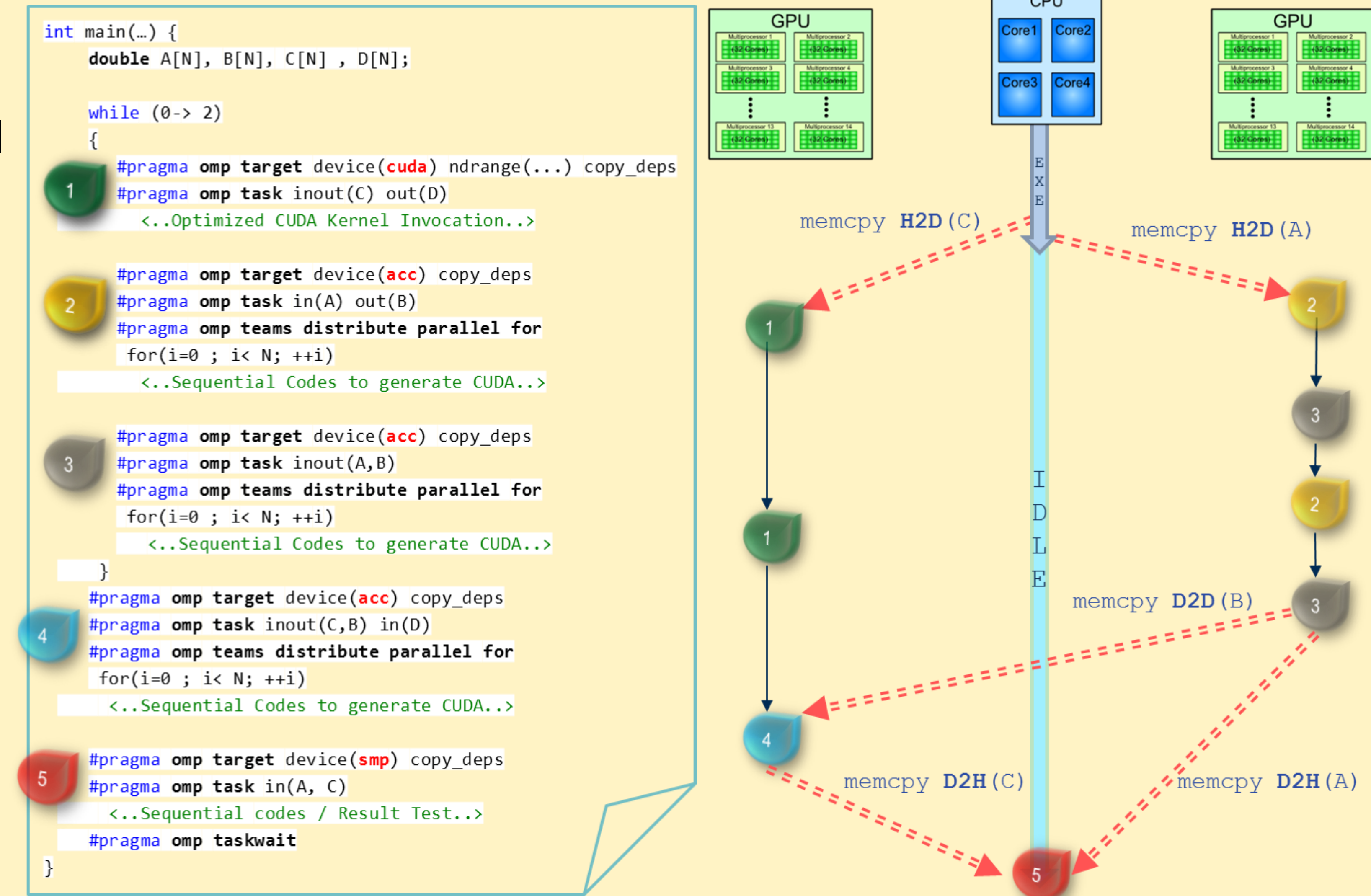
Developed on top of OmpSs model

Mercurium compiler

Source-to-source code generation for C/C++

Targeting NVIDIA GPUs

(Targeting OpenCL as well but not in thesis)



Multi-target Task Share

Problem: Heterogeneity is everywhere. How to exploit entire system?

Solution: We developed multiple target task sharing

