Björn Meyer, Christian Plessl and Jens Förstner

Transformation of Scientific Algorithms to Parallel Computing Code:
Single GPU and MPI multi GPU Backends with Subdomain Support

SAAHPC 2011, 19.07.2011

Motivation and Approach

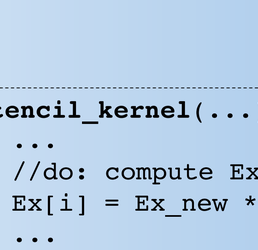
- Accelerate **stencil computation** (nearest neighbors used for computation) algorithms:

Image Filter | Option Pricing | Nanophotonics | Fluid Dynamics

- Algorithms share the property of **regular memory access pattern**
- Allow **transparent usage of accelerators** by domain experts
- **Support multiple architectures** e.g.: Multi-Cores, GPUs, FPGAs, …
- **Separating the algorithm** description **from mapping to** hardware **architecture** (by high-level language (DSL))
- Introducing the novel **concept of domains and subdomains**
- **Efficient code generation by** input and output **representation**

Domain and Subdomain concept

- Real world problems consist of **regions with different properties**
- **Discretization** of continuous **problem space**
- **Mapping** of regions to **domains and subdomains** with arbitrary:
  - Geometry
  - Position in space
  - Operations
  - Properties (e.g. Constants)

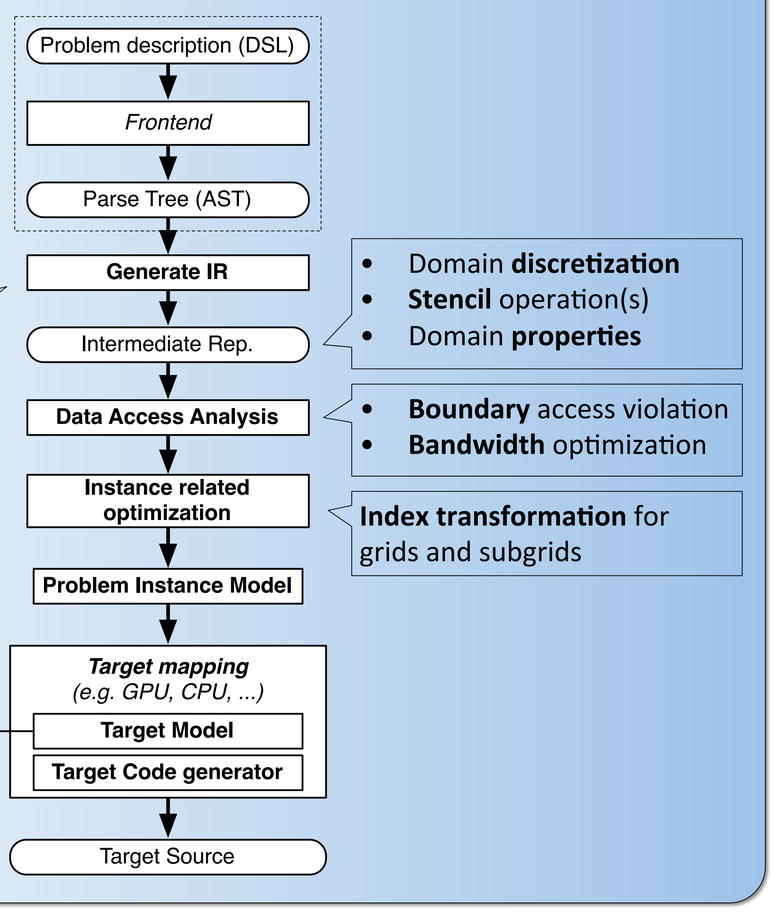


- **Subdomains** have a **hierarchically relationship** to domains
  - Share properties with parent domain
- **Modeling** of **border properties** by domain concept
- **Example**: micro disk cavity in a perfect metallic environment

Code generation tool flow

- **Separate algorithm** from target **mapping**
- Introduce **domain concept** by DSL:
  - Stencil operations
  - Domain properties

- Reduce effort to support new languages and APIs
- Currently supported:
  - CUDA
  - MPI with CUDA



- Domain **discretization**
- **Stencil** operation(s)
- Domain **properties**

- **Boundary** access violation
- **Bandwidth** optimization

Index transformation for grids and subgrids

Generate options:
- **Domain mapping**
  - No mask
  - Multiplicative mask
  - Conditional mask
- CUDA **runtime setup** opt.

Domain mapping Optimizations

No mask
- Geometry **modeled by Function**(x,y)
- **Evaluation at run time**
- Compact representation
  - Reduce needed memory
  - Save bandwidth
- May introduce branch divergence

```
Stencil_kernel(...) {
   ...
   if(pointInDomain(x,y))
     //compute update
   ...
}
```

Multiplicative mask
- Look-up table for each grid point
- Update equation term selected by multiplication
- *Penalties*:
  - Float multiplication
  - Increase needed memory
  - Require additional bandwidth
- No branch divergence

```
Stencil_kernel(...) {
   ...
   //do: compute Ex_new
   Ex[i] = Ex_new * mask[i]
   ...
}
```

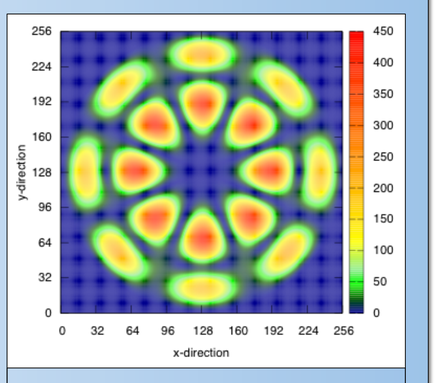
Conditional mask
- Boolean look-up and *IF*
- May Branch divergence

```
Stencil_kernel(...) {
   ...
   if(mask[i])
     //compute update
   ...
}
```

Real-world example

- Realistic **nanophotonic device**
  - **Vacuum microdisk** cavity enclosed **in a perfect metallic environment** (two subdomains)
  - Point-like time-dependent inhomogeneity (optical dipole **point source**)
- Using **Maxwell PDEs** to describe evolution of electromagnetic fields
- Numerical approximation by Finite-Difference Time-Domain method (**FDTD**)
- Known **analytic solution** (Whispering Gallery Modes)
- **Modeling** of different materials **by subdomain concept**

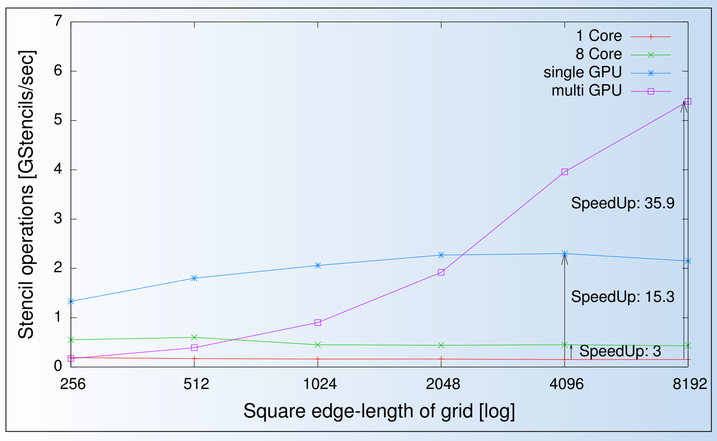
$$
\begin{aligned}
E_x[i] &= ca \cdot E_x[i] + cb \cdot (H_z[i] - H_z[i - dy]) && (1)\\
E_y[i] &= ca \cdot E_y[i] + cb \cdot (H_z[i - dx] - H_z[i]) && (2)\\
H_z[i] &= da \cdot H_z[i] + db \cdot && (3)\\
&\quad \cdot (E_x[i + dy] - E_x[i] + E_y[i] - E_y[i + dx])
\end{aligned}
$$



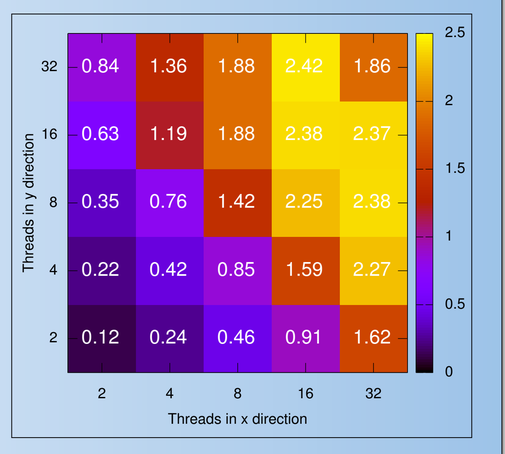
Computed **energy density**

Results

- **Compare reference** CPU **implementation** to:
  - Generated single GPU code (speedup: 15.3x)
  - Generated multi GPU code (speedup: 35.9x)
- Domain **mapping options**
  - **No mask** (2060 MStencils/sec)
  - **Multiplicative mask** (2300 MStencils/sec)
  - **Conditional mask** (2430 MStencils/sec)



SpeedUP and **GStencils/sec over grid size**



Runtime setup variation

Outlook / Future Work

- Support more novel hardware architectures
- Optimization strategies for complex memory access pattern

Contact

Björn Meyer
Department of Computer Science
University of Paderborn

E-Mail: bjoern.meyer@uni-paderborn.de
Phone: +49 5251 604343