Technische Universität Braunschweig

Dynamic parallelism using CUDA

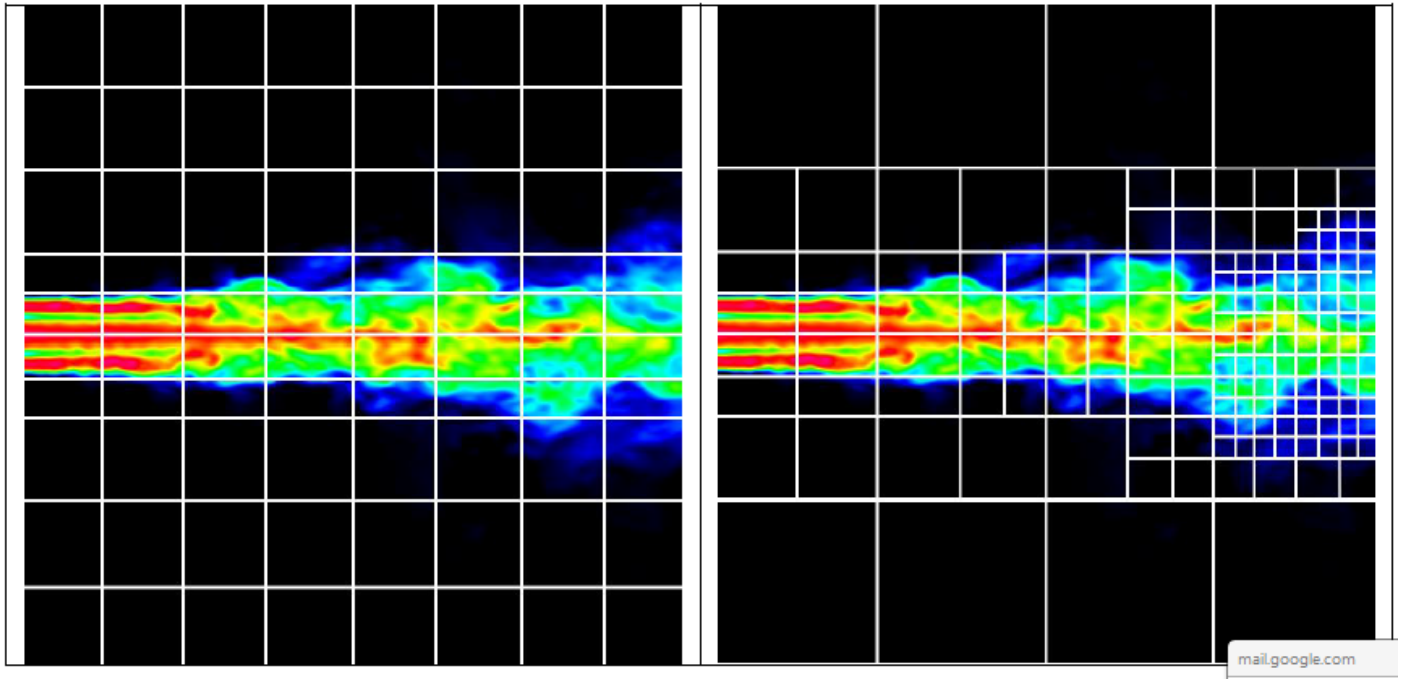
Example by shallow water equation

Marc Kassubeck, Torsten Thoben, 17. März 2015

Inhalt

Explaining the Task

- extend an existing shallow water sovler
- with dynamic prallelism (DP)



Implementation DP with CUDA

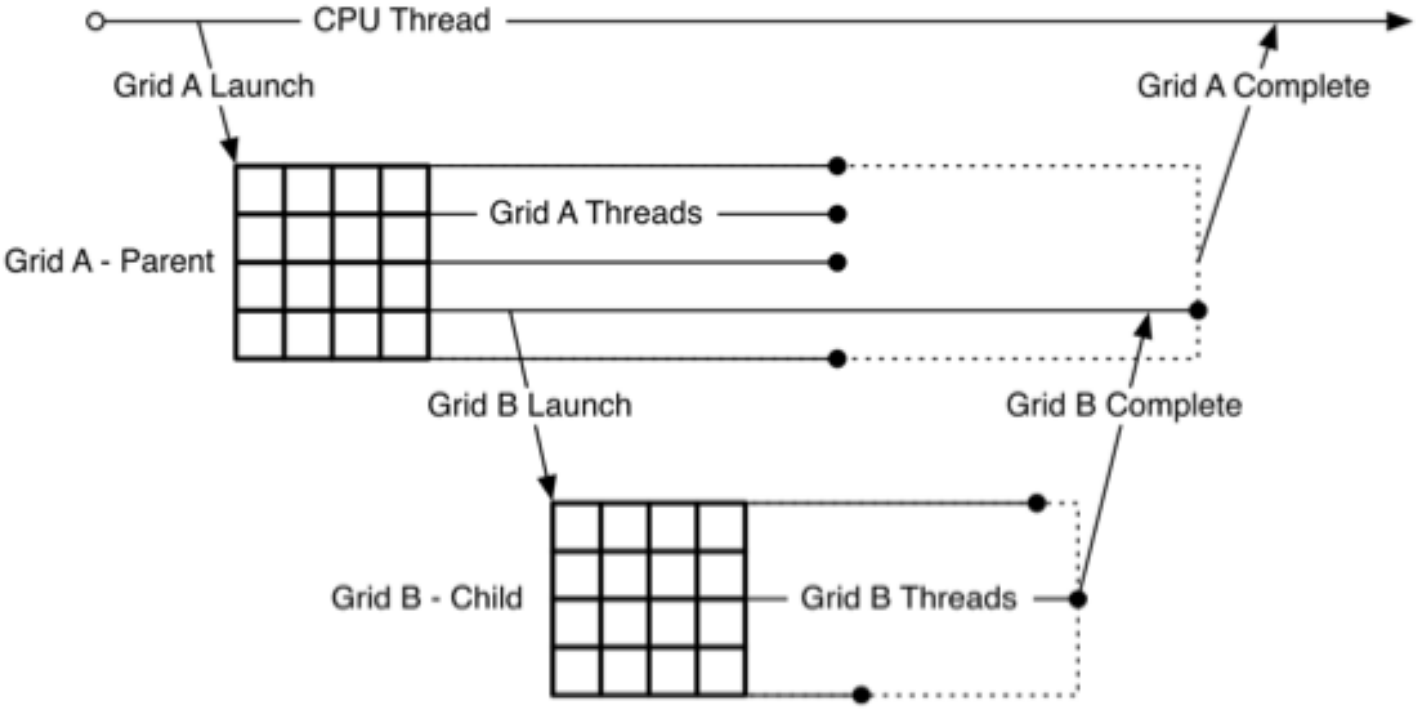
- DP in CUDA means calling a Kernel in a Kernal
- so every thread calls a new grid

```
main(){
    ...
    <<< g,b >>> func(depth)
}

__global__ void func(int depth)
{
    if(depth<2)
        func  <<< g, b >>> (depth+1)
}
```

DP and Syncronisation

- One Kernel finished if his subkernel finished but it can work on after the supkernel call.
- To syncronize between one thread and his subkernels use cudaDeviceSynchronize() (this will not syncronize between threads)



Recursion Depth and Device Limits

- nesting depth, limit to 24
  - need to reserver buffer for running/suspended or launching kernels
  - cudaDeviceSetLimit(cudaLimitDevRuntimePendingLaunchCount, x);
  - default is set to 2048
- synchronization depth
  - cudaDeviceLimit(cudaLimitDevRuntimeSyncDepth, x)
  - standart is 2

Problem: Implemting of the datastructure

- Datastructure: Forest (many Trees)
- Limit to getting coarse is the Forestsize (number of Trees)
- Limit to getting fine end up to the maximum recursions
- Trees: eg Quadtrees, Octrees...

With Pointer

```
class LeafElem : public TreeElem
{
public:
    float value;
    __device__ __host__ LeafElem();
    virtual __device__ __host__ ~LeafElem(){};
    virtual __device__ __host__ bool isLeaf()
    {
      return true;
    }
};
```

With Pointer

```
class BranchElem : public TreeElem
{
 public:
    int nx;
    int ny;
    int depth;
    TreeElem** children;
    __device__ __host__ BranchElem(int nx, int ny, int depth)
    virtual __device__ __host__ ~BranchElem();
    virtual __device__ __host__ bool isLeaf()
    {
        return false;
    }
};
```

But this dont work

Storing the Tree-Structure in an Array

0	0	0	0	1	1	2	2
0	0	0	0	1	1	2	2
0	0	0	0	1	1	1	1
0	0	0	0	1	1	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0