Preparing the Property Evaluation in OPM Flow for GPU Execution

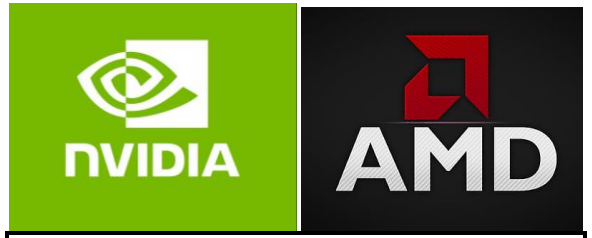
Tobias Meyer Andersen

SINTEF Digital

OPM Summit 2025 Bergen, May 27th

Main topics

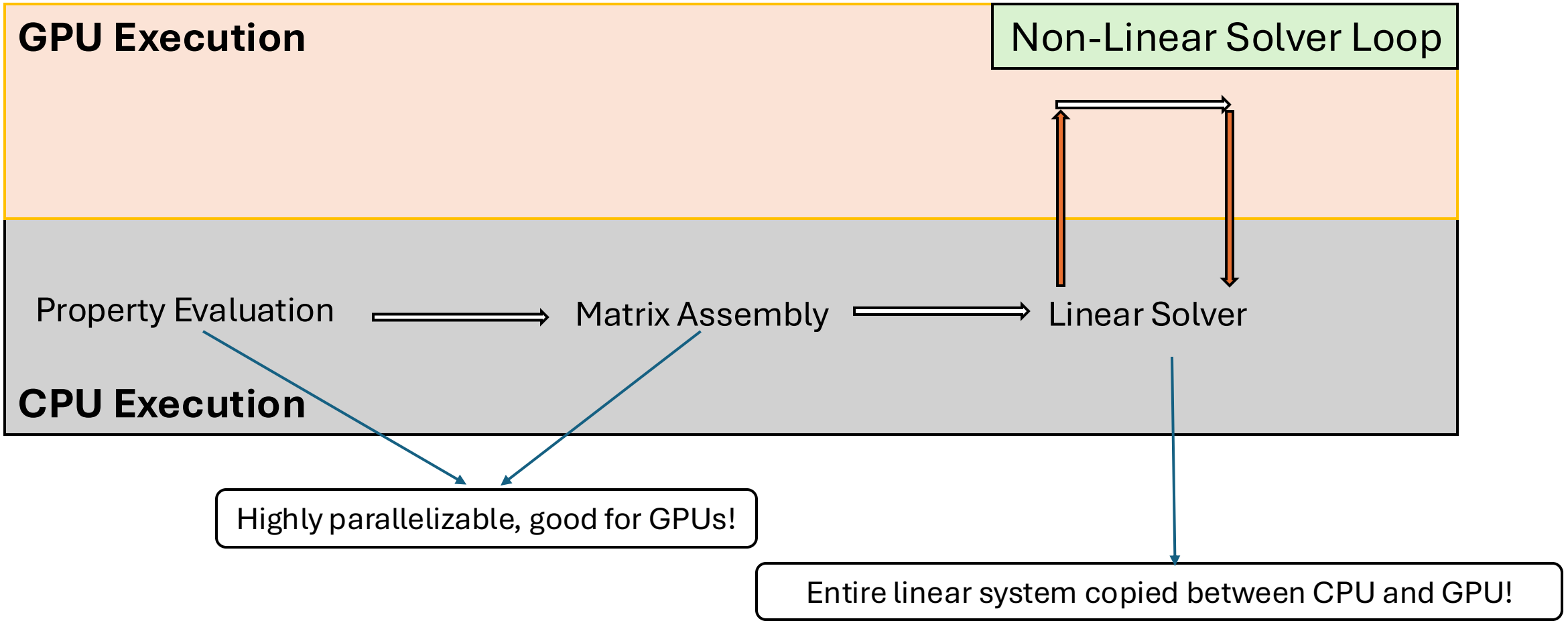
- This talk is mainly aimed at developers and maintainers of OPM
- Familiarize OPM developers with new changes in the code
  - Context of GPU work
  - Introduction to GPU programming and memory
  - Show new and recurring design patterns
  - Example of rewriting simple class
- Future work



CUDA and **HIP** are Nvidia and AMDs language for programming GPUs. They are very similar and OPM supports both

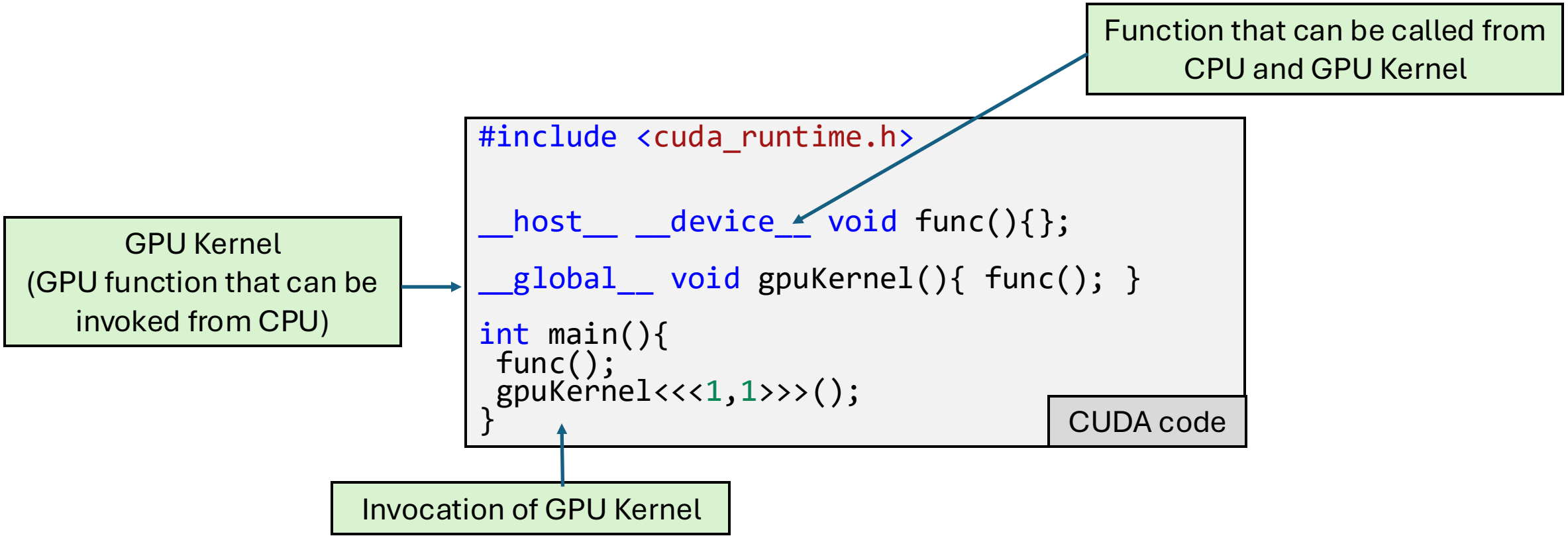
OPM Flow on GPU Status

- Currently only the linear solver can run on the GPU



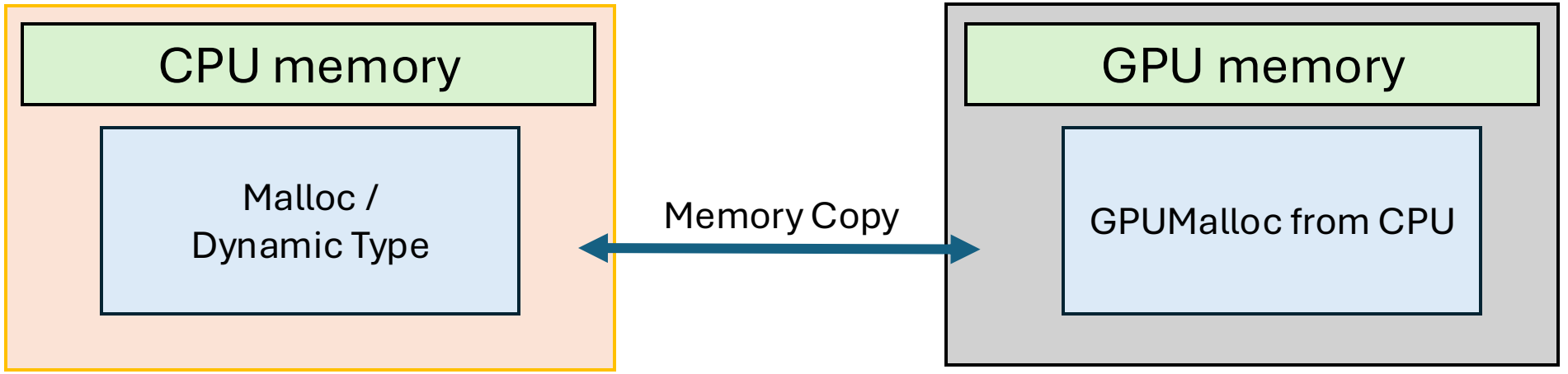
Introduction to GPU code terminology

A GPU kernel is a function called from the CPU, which launches a parallel workload on the GPU



Dynamic memory model

- GPU and CPU have separate memory space
  - Explicit copies of memory buffers between the two!
- C-style memory management, malloc & free



- Types with known amount of memory usage can be put on stack
  - This means we can pass by value to GPU!

Managing GPU and CPU in Kernels

- Memory management is not allowed inside a kernel
  - Smart pointers are not supported either
- The trichotomy below describes dynamic objects
  - inspired by CUDA.jl and KernelAbstractions.jl
- Transform the CPU property objects with these two steps:

CPU dynamic type	**GPU Buffer**	**GPU View**
Std::vector etc… Everything is allowed	Owns and handles GPU memory Can be of dynamic size Can use smart pointers **Cannot** be sent into GPU kernel	Only a pointer to GPU memory Cannot use smart pointers **Can** be sent into GPU Kernel

copy_to_gpu<…>(CPUType)

make_view<…>(GPUBufType)

Avoiding static non-constexpr

- Statically allocated memory is not well defined on GPUs
  - Many static classes in OPM-Common, we have rewritten at least:
    - BlackOilFluidSystem – made dynamic with clever preprocessing tricks by Kjetil O. Lye
    - CO2Tables
- Const-expr works fine because the compiler inserts actual raw value

Example of Making Class GPU Friendly

- Simple interpolation class using std::vector

```
class LinearInterpolationClass
{
   std::vector<> data;
   void someFunction();
}
```

C++ pseudocode

- This class cannot be used inside a GPU function
  - Std::vector<> is not allowed!

Example of Making Class GPU Friendly

- GPU Version of that class will look like this:

Add template argument with default value

```
template <class VectorT = std::vector<>>
class LinearInterpolationClass
{
   VectorT data;
   OPM_HOST_DEVICE void someFunction();
}

template<...> auto copy_to_gpu<...>(cpuObject);
template<...> auto make_view<...>(gpuBufObject);
```

GPU macro compiling this function for GPU and CPU

C++ pseudocode

- copy_to_gpu takes regular version and creates **<GpuBuffer<>>**
- make_view takes in the buffer version and makes a **<GpuView<>>**
- **OPM_HOST_DEVICE** is "__host__ __device__" when compiling for GPU
  - Macros like **OPM_IS_INSIDE_DEVICE_FUNCTION** specialize implementations

DualBuffer

- Can a buffer contain buffered types?
  - GpuBuffer of GpuBuffer would mean GpuView must be dynamic
  - If it contains GpuBuffer of Views, then who owns what the views view?
- DualBuffer is a version of GpuBuffer that stores **two** buffers
  - Std::vector of the inner buffers
  - A buffer of GpuViews of the inner type
    - The inner elements are owned by the vector
    - Making a view of the buffer is just providing the pointer!

CPU dynamic type	**Buffer version**	**Kernel/View version**
Std::vector<std::vector<>>	Std::vector<GpuBuffer<>> GpuBuffer<GpuView<>>	GpuView<GpuView<>>

Current Progress

- Working towards SPE11C proof-of-concept simulation
- Families of classes now supported on GPU
  - Interpolation/AD classes
  - BlackOilFluidSystem
  - PVT classes
  - Material laws
  - FlowProblem
- This is pretty much everything needed for the proof-of-concept

Future work

- Make proof-of-concept matrix assembly on GPU
  - We can then run the entire non-linear solver (no wells) on GPU!
- Detailed profiling to optimize memory and performance
- Expand support of physics in property evaluation
  - Run more complex cases than SPE11C…

Acknowledgement

We thank Equinor for funding this work

Contact: tobiasmeyer.andersen@sintef.no

SINTEF