

# New Developments for Linear Solvers within OPM's GPU-ISTL Framework

**Jakob Torben**

Tobias Meyer Andersen, Kjetil Olsen Lye, Atgeirr Rasmussen,  
Olav Møyner, Arne Morten Kvarving Halvor Nilsen

SINTEF Digital



# Main topics



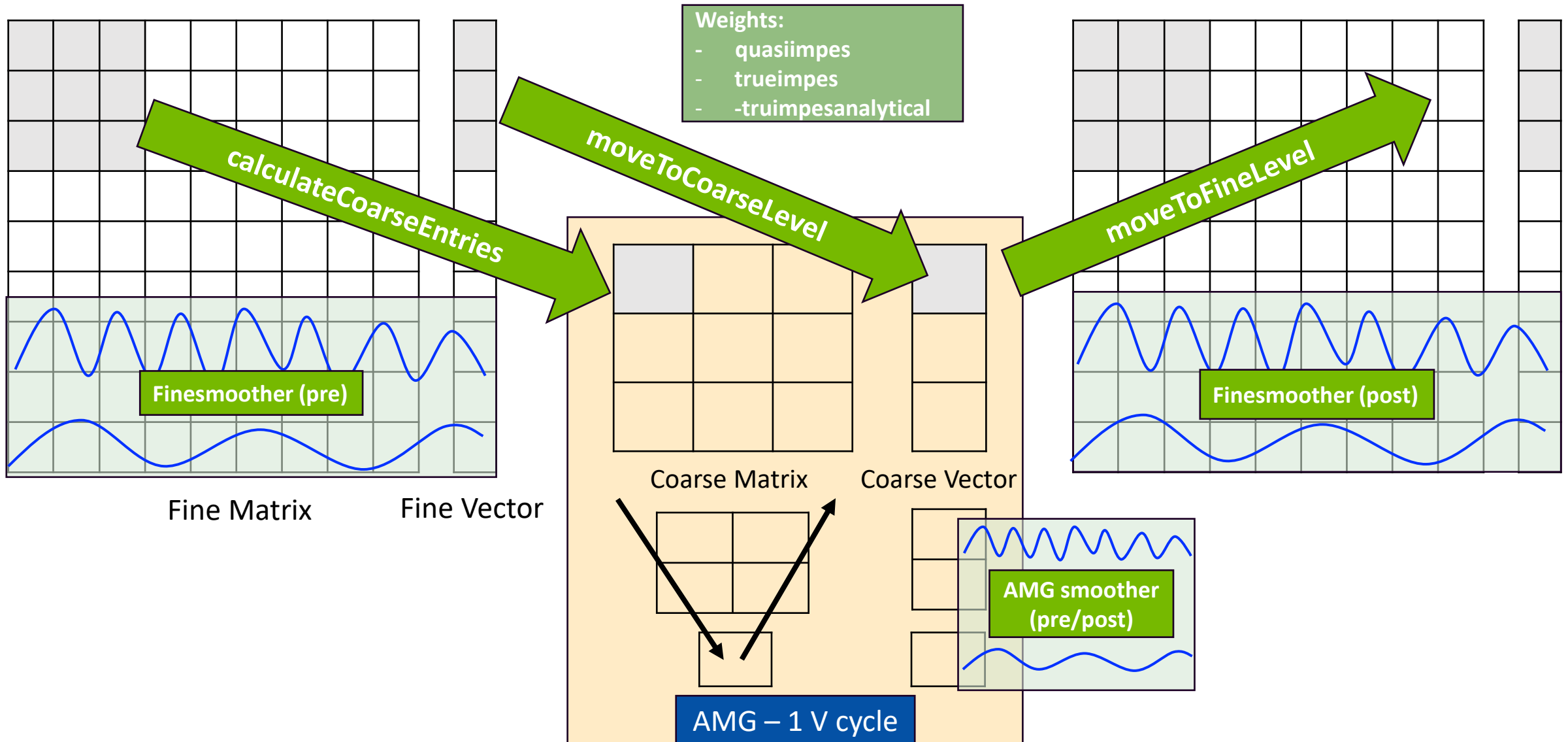
- **Background:** Linear solvers in OPM, CPR-AMG and Gpu-ISTL
- **Updates on ILU0/DILU:** Progress since last time
- **New AMG Solvers:** HYPRE & AMGX integration
- **GPU as First-Class Citizen:** Backend=cpu/gpu & factory updates
- **CPR-AMG on the GPU:** Overview of implementation
- **Early Benchmark Results:** Norne, SPE10 and Asset Model
- **Future Work:** Optimisation, extensions, and more

# Background: GPU Acceleration of Linear Solvers with Gpu-ISTL



- Linear solver is the cornerstone of a performant reservoir simulator
  - Natural place to start GPU acceleration
- Promising results on our implementation of ILU0/DILU preconditioners on the GPU
- **CPR-AMG:** The default is the Constrained Pressure Residual (CPR) method with an Algebraic Multigrid (AMG) preconditioner
  - **CPR:** A two-stage preconditioner. It solves an approximate pressure system (coarse problem) and then applies a smoother to the full system (fine problem correction)
  - **AMG:** Solves the coarse pressure system efficiently. It works by:
    - **Aggregation/Classical:** Building a hierarchy of coarser grids based only on the matrix structure
    - **Smoothing and Interpolation:** Solving on coarse grids captures low-frequency errors, while smoothers handle high-frequency ones; interpolation moves solutions between levels
- **Goal:**
  - GPU implementation of CPR-AMG that is faster than MPI parallel CPU implementation on relevant cases
  - Maintainable and reuses as much of the code and strategy of CPU implementation as possible
  - Supports NVIDIA and AMD GPU hardware

# Background: CPR-AMG



# Update on GPU ILU and DILU preconditioners

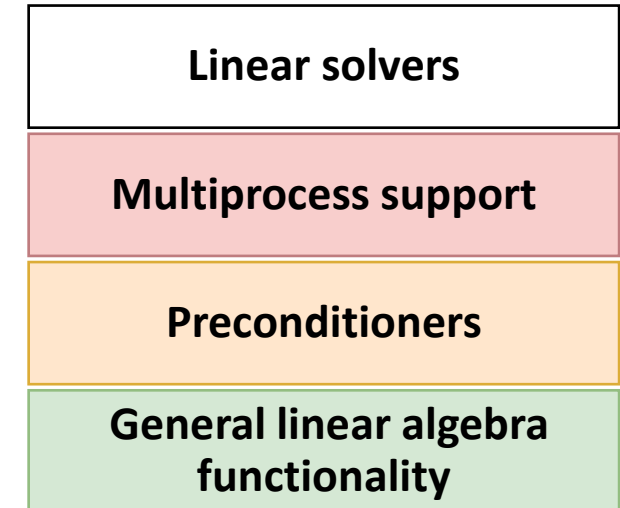
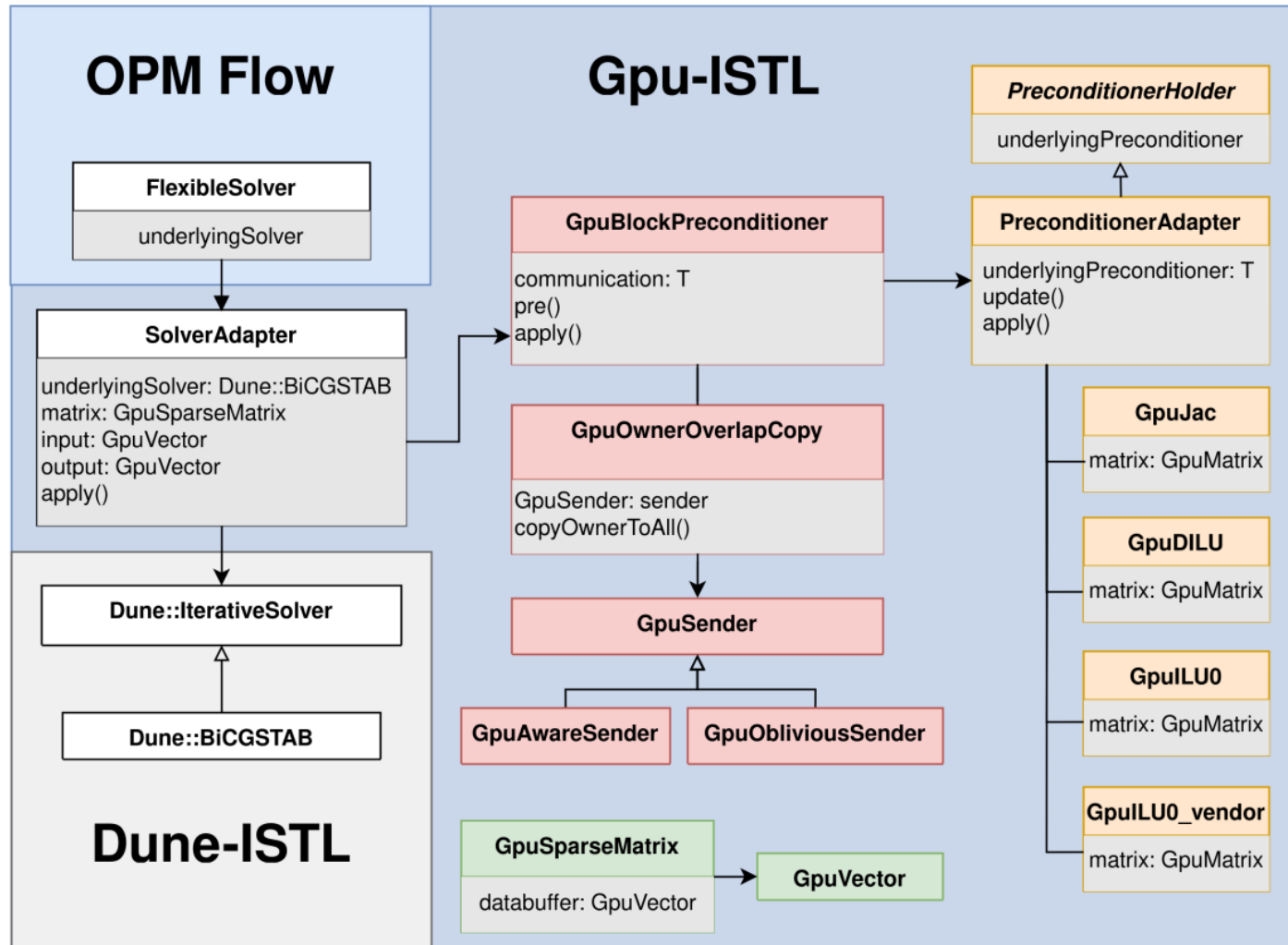


- ILU(0) has also been implemented in Gpu-ISTL
- DILU has obtained significant speedup due to
  - New autotuning step
  - New underlying data structures (matrix splitting)
  - Combining function calls with CUDA/HIP graphs
  - Adding mixed precision schemes that do not increase iteration count
- Last year 1.17 speedup over CPU CPR was presented on SPE11C
  - Now closer to 2.5
- Last year 2M cells were needed for GPU > CPU on SPE11C
  - Now only 270k cells are needed!



# Gpu-ISTL

The Gpu-ISTL framework provides Dune-ISTL compatible sparse linear operations on the GPU for OPM Flow



Lye et al., (2025). gpu-ISTL - Extending OPM Flow with GPU Linear Solvers. Journal of Open Source Software, 10(109), 7740, <https://doi.org/10.21105/joss.07740>

# New AMG solvers: HYPRE



- HYPRE: High-performance preconditioners, including BoomerAMG
- Developed by Lawrence Livermore National Laboratory
- Highly active (12 332 commits, last commit 5 days ago)
- OPM Integration:
  - Implemented as a preconditioner within OPM
  - Supports both CPU and GPU execution (CUDA & HIP)
  - Currently focused on BoomerAMG and scalar matrices for AMG in CPR-AMG
  - Internally handles data transfer between Dune/OPM and HYPRE objects
- Current status:
  - BoomerAMG solver is available for CPU and GPU, with MPI parallelism planned
  - Implementation does not yet support GPU to GPU transfers between Gpu-ISTL and HYPRE objects



# New AMG solvers: AMGX



- AMGX: NVIDIA's high-performance AMG library for GPUs
- Limited activity (265 commits, last commit 3 months ago...)
- OPM Integration:
  - Implemented as a preconditioner, supporting GPU execution
  - Added support for scalar matrices and AMG solver
  - Internally handles data transfer to/from AMGX memory
  - Allows direct GPU-to-GPU transfers when used with Gpu-ISTL solvers



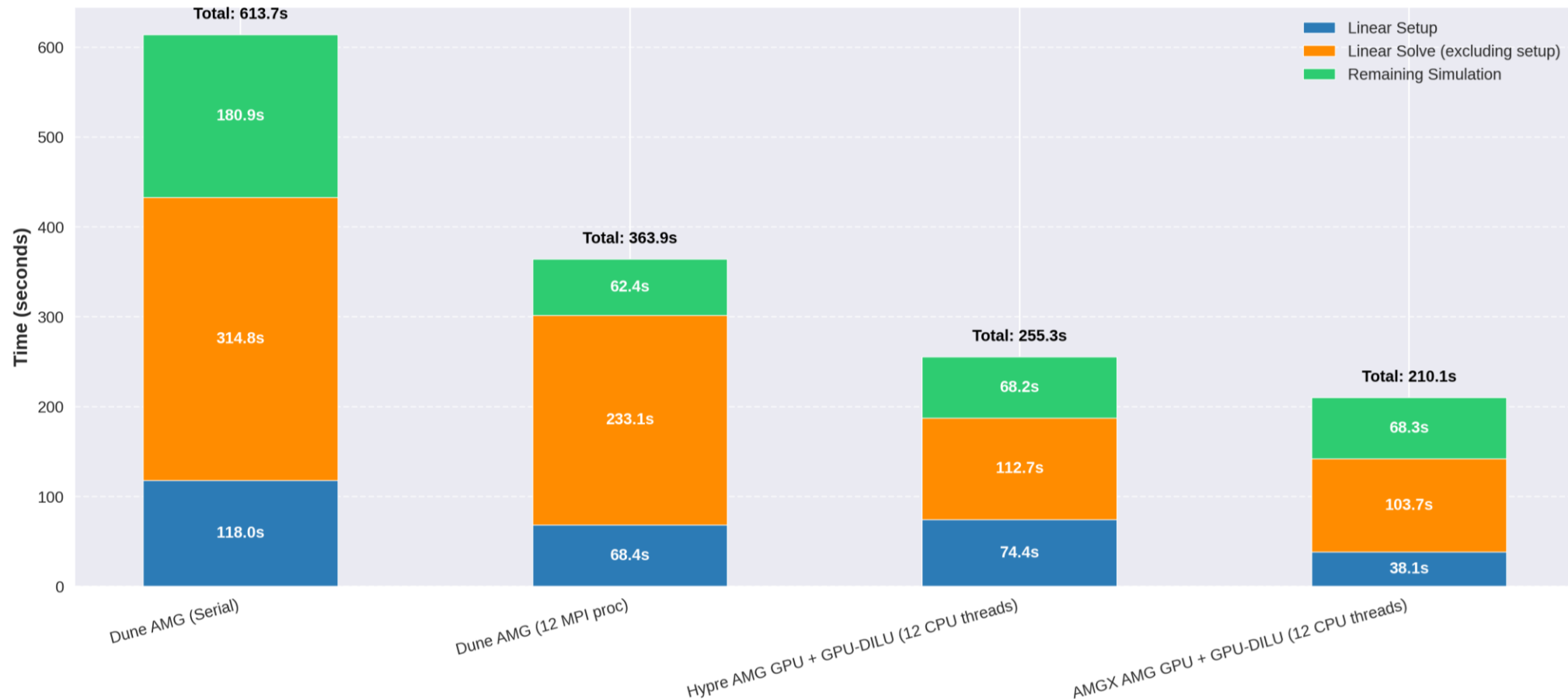
**NVIDIA**®



# Results from December 2024 - SPE10



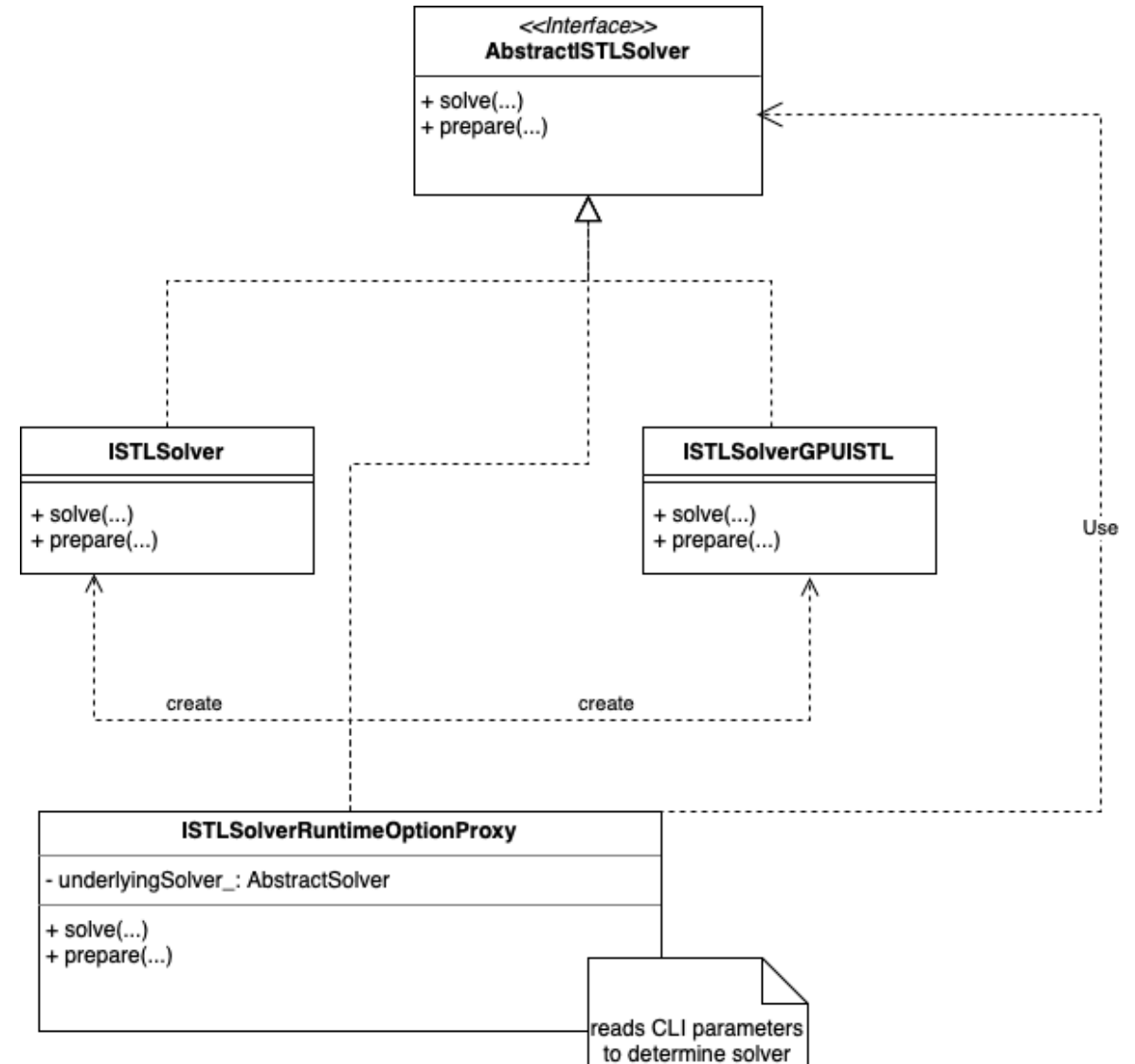
Number of cells: 1M



**CPR-AMG not fully on GPU, copying up and down from CPU!**

# GPU as a First-Class Citizen in OPM

- Previous Gpu-ISTL was hidden away:
  - Extra memory copies needed
  - Convoluted design for accessing matrix
- PR #6270:
  - We handle GPUs at the ISTLSolver level
  - New CLI argument:
    - `--linear-solver-accelerator={gpu,cpu}`



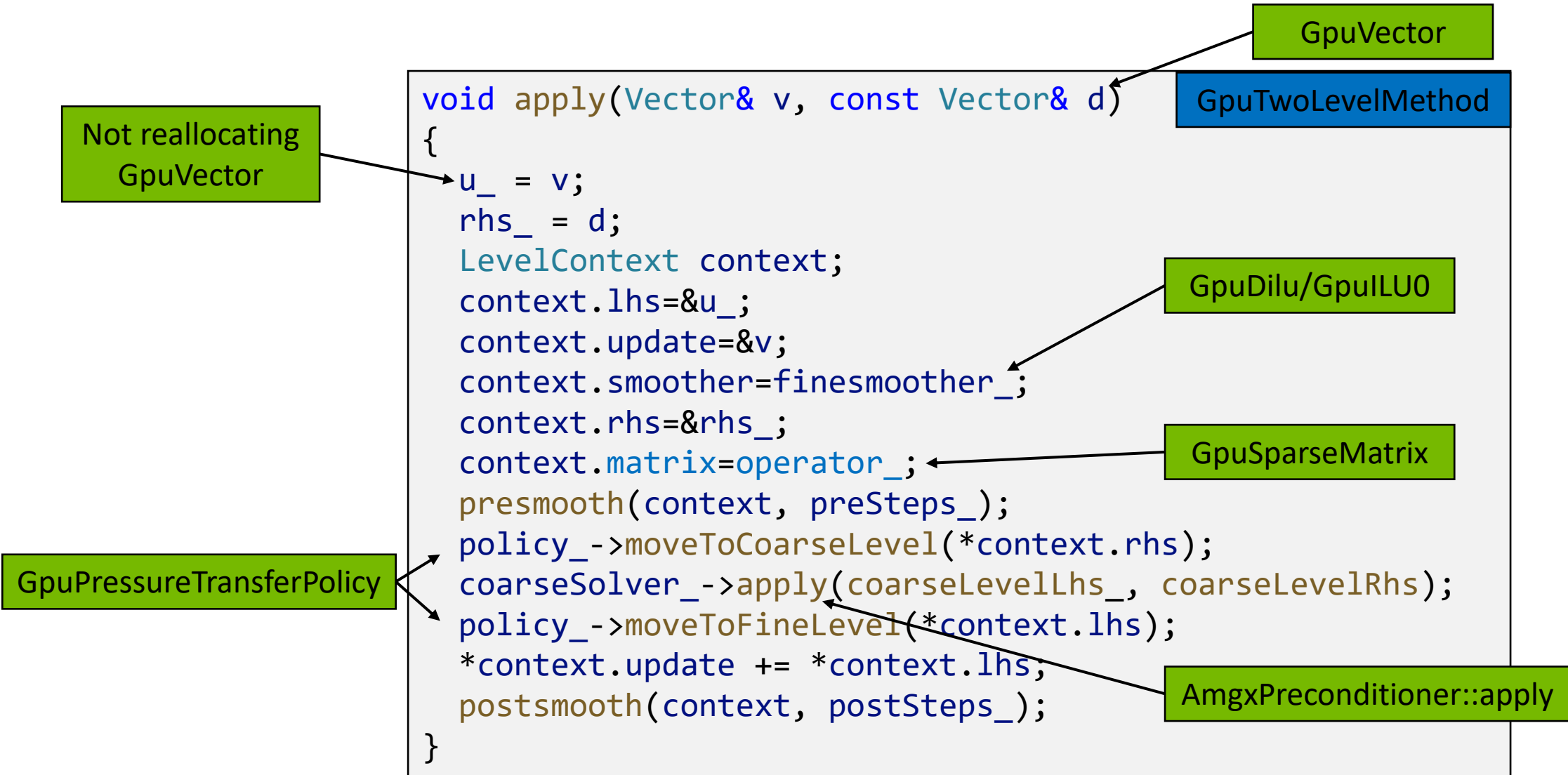
Kjetil Olsen Lye

# CPR-AMG on GPU: Overview



- **PR #6308:**
  - Implementation of CPR-AMG preconditioner running entirely on the GPU using Gpu-ISTL
- **Core CPR operations on GPU:** Native GPU implementations for:
  - `moveToCoarseLevel`
  - `moveToFineLevel`
  - `calculateCoarseEntries`
- **Weights Calculation:**
  - `quasiimpes`: Fast, native GPU implementation
  - `trueimpes` / `trueimpesanalytic`: Still CPU-based but results copied to GPU (often worth the copy due to fewer iterations)
- **GPU Resource Management:**
  - Uses a single GPU matrix for the linear solver, preconditioner and matrix operations, minimising memory use and transfers
  - Supports direct GPU-to-GPU transfers when using AMGX
- `GpuSparseMatrix` extended to support block size 1 for coarse matrix operations
- **Code Structure:** Near-identical code for CPU (`twoLevelMethod`) and GPU (`gpuTwoLevelMethod`), aiming for full unification
- **Testing:**
  - New GPU-specific tests added (`AmgxInterface`, `GpuPressureTransferPolicy`, etc.)
  - Good consistency observed for CPR operations between CPU and GPU runs
  - Tested on various cases, showing good convergence (often better than CPU)

# CPR-AMG on GPU: Implementation Details



# CPR-AMG on GPU: Implementation Details cont.



```
void presmooth(LevelContext& levelContext, size_t steps)
{
    for(std::size_t i=0; i < steps; ++i) {
        *levelContext.lhs=0;
        *levelContext.smoother.apply(*levelContext.lhs, *levelContext.rhs));
        // Accumulate update
        *levelContext.update += *levelContext.lhs;
        // update defect (_A_ -> usmv(alpha,x,y))
        levelContext.matrix->appliescaleadd(-1, *levelContext.lhs, *levelContext.rhs);
        levelContext.pinfo->project(*levelContext.rhs);
    }
}
```

smoother.hh (Dune-ISTL)

GpuDilu::apply

GpuSparseMatrix mv (cuSPARSE) call

GpuVector += (cuSPARSE) call

# CPR-AMG on GPU: CLI Command Specifies Implementation



- Same linear solver specification for CPU and GPU
- CLI option chooses implementation

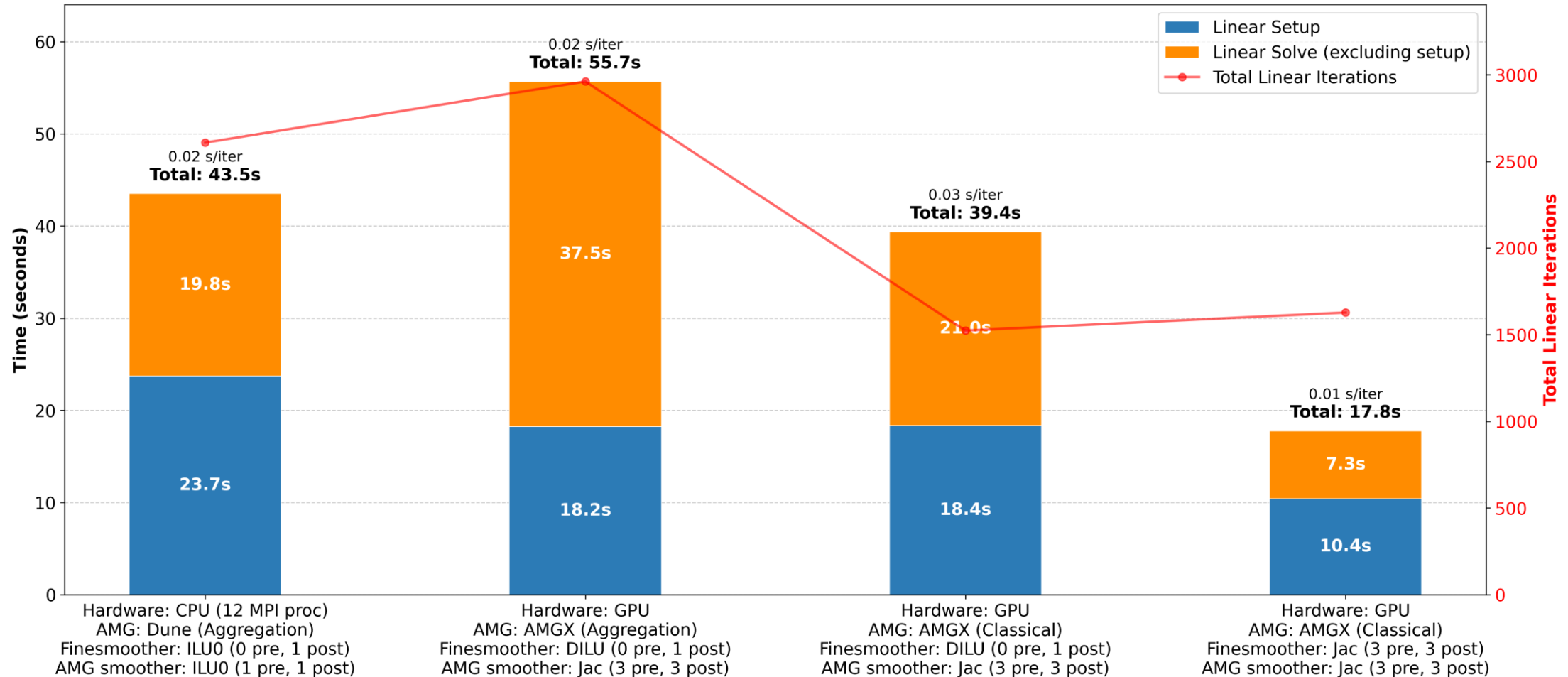
```
--linear-solver-accelerator={gpu,cpu}
```

- Flexible solver factory (bicgstab, loopsolver)
  - Operator and vector types is either CPU or GPU matrix
- Preconditioner factory
  - **Serial CPU**
    - "dilu": DILU CPU implementation
    - "gpudilu": GpuDILU with preconditioner adapter
    - "amgx": AmgxPreconditioner with CPU types
  - **MPI CPU**
    - "dilu": DILU CPU implementation wrapped as blockPreconditioner
    - "gpudilu": GpuDilu with PreconditionerAdapter and GpuBlockPreconditioner
  - **Serial GPU**
    - "dilu": GpuDILU
    - "amgx": AmgxPreconditioner with GPU types
  - **MPI GPU**
    - Implementation in progress
- ISTLSolver/ISTLSolverGPU/ISTL chooses appropriate "weight\_type" implementation

```
{
  "maxiter": "20",
  "tol": "0.005",
  "verbosity": "0",
  "solver": "bicgstab",
  "preconditioner": {
    "type": "cpr",
    "weight_type": "trueimpes",
    "pre_smooth": "0",
    "post_smooth": "1",
    "finesmoother": {
      "type": "dilu"
    },
    "verbosity": "0",
    "coarsesolver": {
      "maxiter": "1",
      "tol": "0.1",
      "solver": "loopsolver",
      "verbosity": "0",
      "preconditioner": {
        "type": "amg/amgx"
      }
    }
  }
}
```

# Early Benchmark Results - Norne

Number of cells: 44k

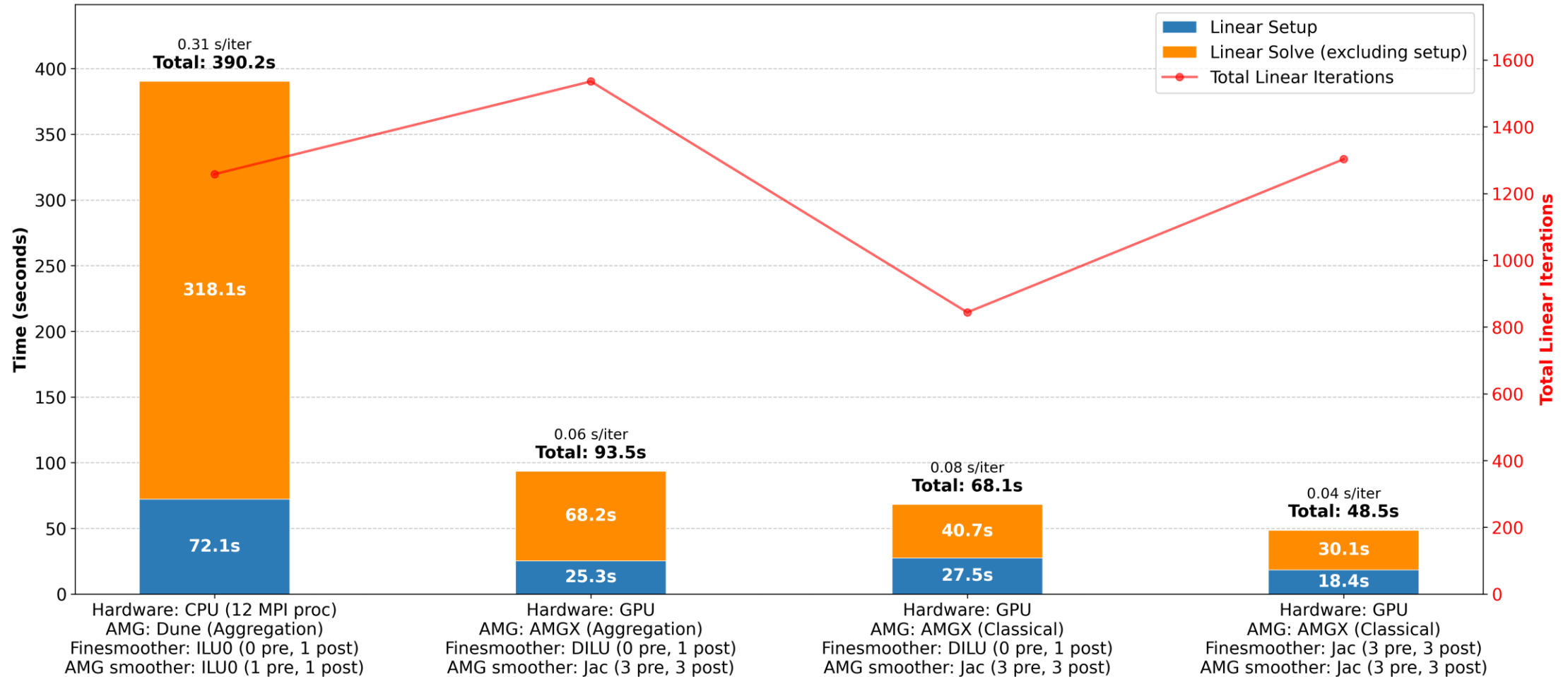


Note: Slide has been revised on (03.06.2025) to reflect latest available test results

# Early Benchmark Results – SPE10



Number of cells: 1M

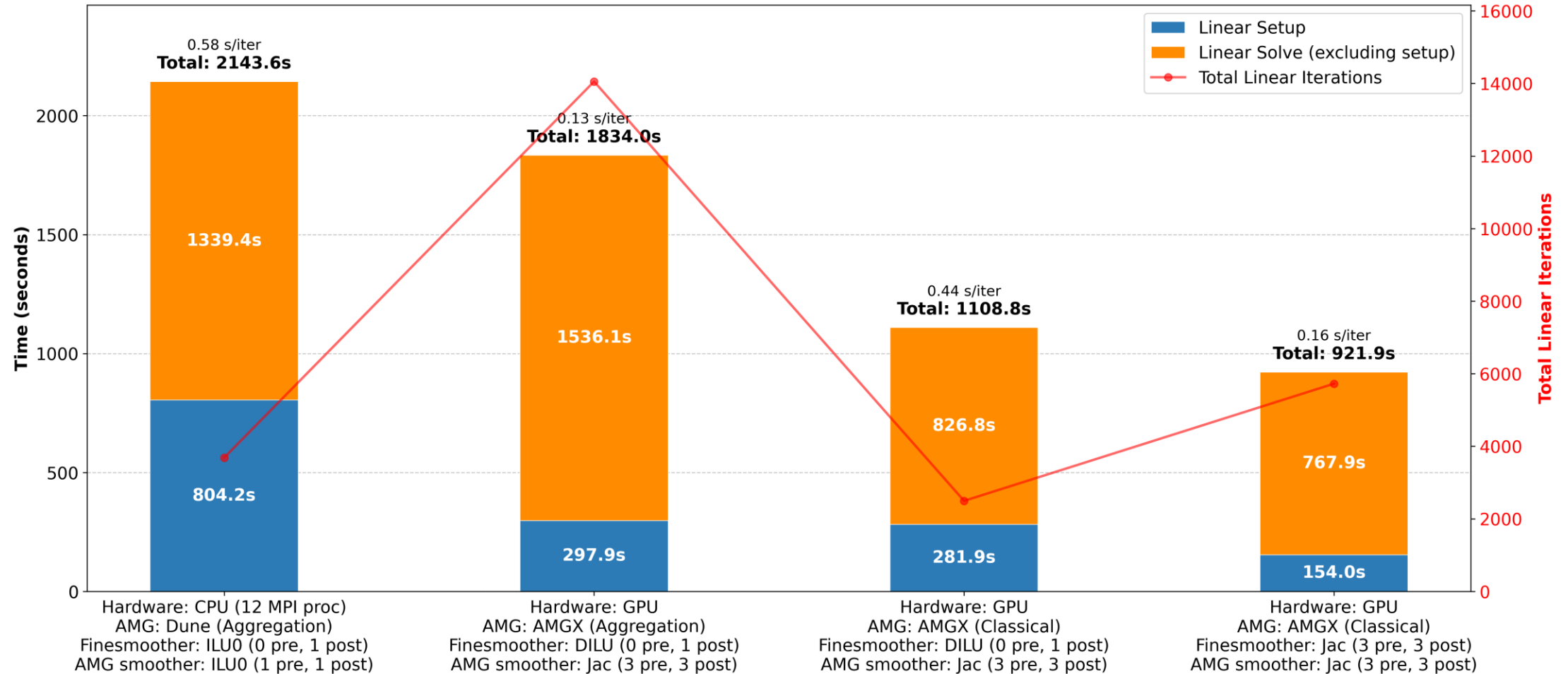


Note: Slide has been revised on (03.06.2025) to reflect latest available test results

# Early Benchmark Results – Asset Model for Real Field



Number of cells: 800k



Note: Slide has been revised on (03.06.2025) to reflect latest available test results

# Ongoing and Future Work



- **Optimisations and Ongoing work:**

- Detailed profiling and optimisations
- Have a closer look at HYPRE again
- Test on AMD GPUs, need AMG solver on AMD hardware:
  - HYPRE, HIPify AMGX, or AMD's rocALUTION AMG solver

- **Further Studies:**

- Test on more and larger cases, including CO2 cases. Test on HPC hardware
- Investigate optimal linear solver parameters (AMG types, interpolators, smoothers, etc.)
- Explore AMG hierarchy reuse strategies for faster setup
- Lessons transferable to the current CPU implementation?

- **Feature Extension:**

- Well operators on the GPU + CPRW preconditioner
- Implement MPI parallelism
  - HYPRE (CPU and GPU), AMGX (GPU)

---

## Acknowledgments:

We thank Equinor for funding this work

Jakob Torben

[jakob.torben@sintef.no](mailto:jakob.torben@sintef.no)

