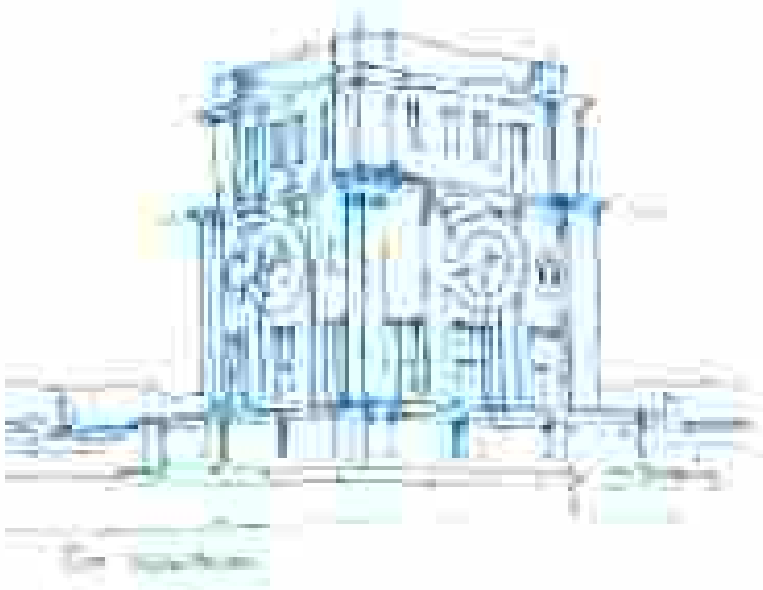
Performance Portability and Evaluation of Heterogeneous Components of SeisSol Targeted to Upcoming Intel HPC GPUs

Computational Sciesmology with SYCL

Ravil Dorozhinskii, Ludwig Krotz, Michael Bader,
Technical University of Munich

14th of November, 2021



oneAPI Dev Summit, Virtual Conference

Outline of talk

Introduction

GPU computing in SeisSol

SYCL Integration into SeisSol

Performance Evaluation

Simulations with oneAPI

Continuous Integration

Conclusion & Future Work

NOTE:

Illustrations and results without annotations are mostly taking from [2] and [1]

What is SeisSol?

SeisSol - software for simulating seismic waves and earthquake dynamic

based on:

- Discontinious Galerking method
- ADER time-integration scheme
- Tetrahedral meshing

supports:

- Elastic and visco-elastic wave propagation models
- Plasticity model
- Local and Global Time-Stepping schemes
- Point sources and rupture surfaces to model source terms
- Fused-simulations

originally came with:

- MPI+OpenMP parallelization
- Code generator - YATeTo DSL. [3]

ADER-DG in a Nutshell

Update Scheme

$$Q_k^{n+1} = Q_k^n + M^{-1}(K^\xi D_\xi A_k^* + K^\eta D_\eta B_k^* + K^\zeta D_\zeta C_k^*) \tag{1}$$

$$- \frac{1}{|J|} M^{-1} \left(\sum_{i=1}^{4} |S_i| F^{-,*} D_\chi \bar{A}_k^i \right)$$

$$- \frac{1}{|J|} M^{-1} \left(\sum_{i=1}^{4} |S_i| F^{-,*} A_{k(i)} D_{k(i)} \bar{A}_{k(i)} \right)$$

Cauchy-Kowalewski

$$\bar{D}_k = \sum_{i=0}^{O-1} \frac{(t^{n+1} - t^n)^{i+1}}{(i+1)!} \frac{\partial^i}{\partial t^i} Q_k^n \tag{2}$$

$$\frac{\partial^{i+1}}{\partial t^{i+1}} Q_k^n = M^{-1} \left[(K^\xi)^T \left(\frac{\partial^i}{\partial t^i} Q_k^n \right) A_k^* + (K^\eta)^T \left(\frac{\partial^i}{\partial t^i} Q_k^n \right) B_k^* + (K^\zeta)^T \left(\frac{\partial^i}{\partial t^i} Q_k^n \right) C_k^* \right] \tag{3}$$

Source Code Structure and Code Generation with YATeTo

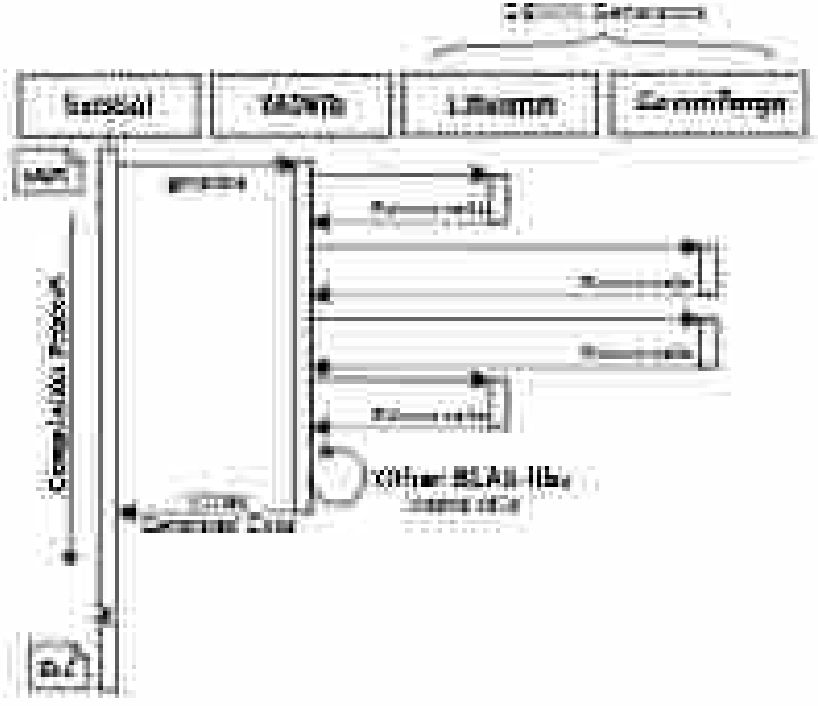


Figure: Compilation process

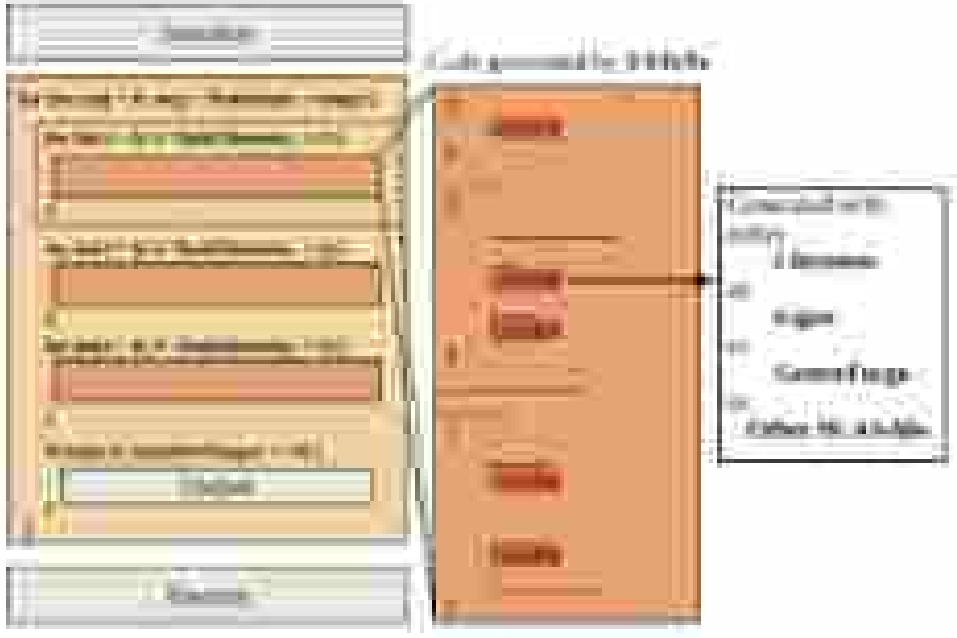


Figure: Simplified source code structure

```
volumeSum = self.Q['kp']
for i in range(3):
  volumeSum += self.db.kXiiM[i][self.t('kl')] * self.I['iq'] * self.starMatrix[i]['qp']

volume = (self.Q['kp'] <= volumeSum)
generator.add('volume', volume)
```

Listing: Example of YATeTo DSL

GPU computing in SeisSol

Figure: CPU/GPU task parallelism

Binary Batched Operations:

- Trivial grid/block distribution
- Easy to estimate run-time resources i.e., shared memory, registers

But:

- Finer granularity w.r.t CPU-like parallelism
- Lower arithmetic intensity

GemmForge on Nvidia V100

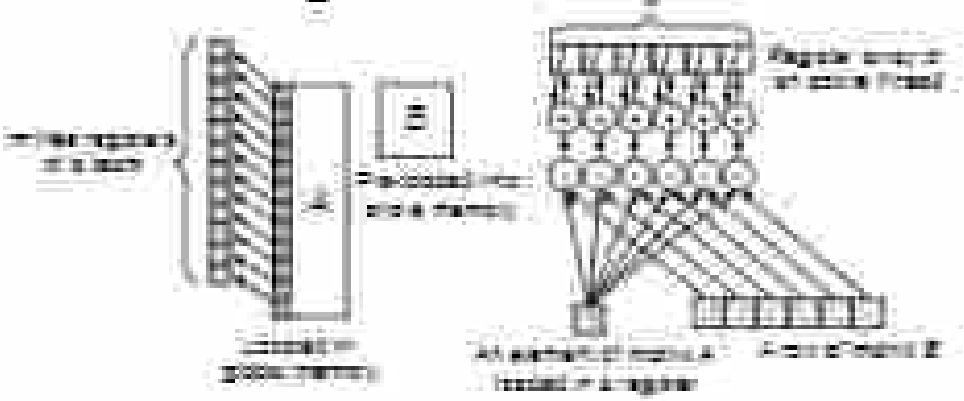
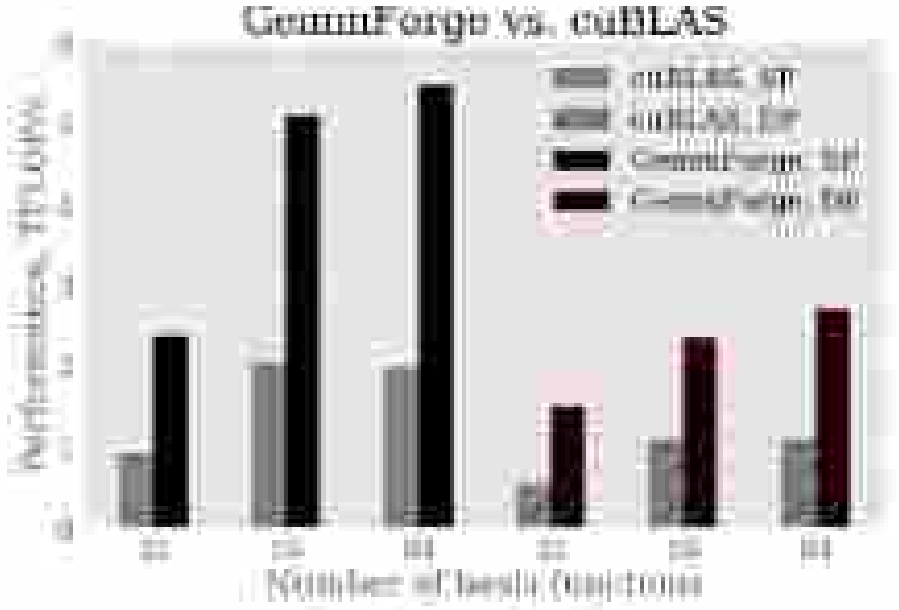


Figure: Sum of parallel outer products



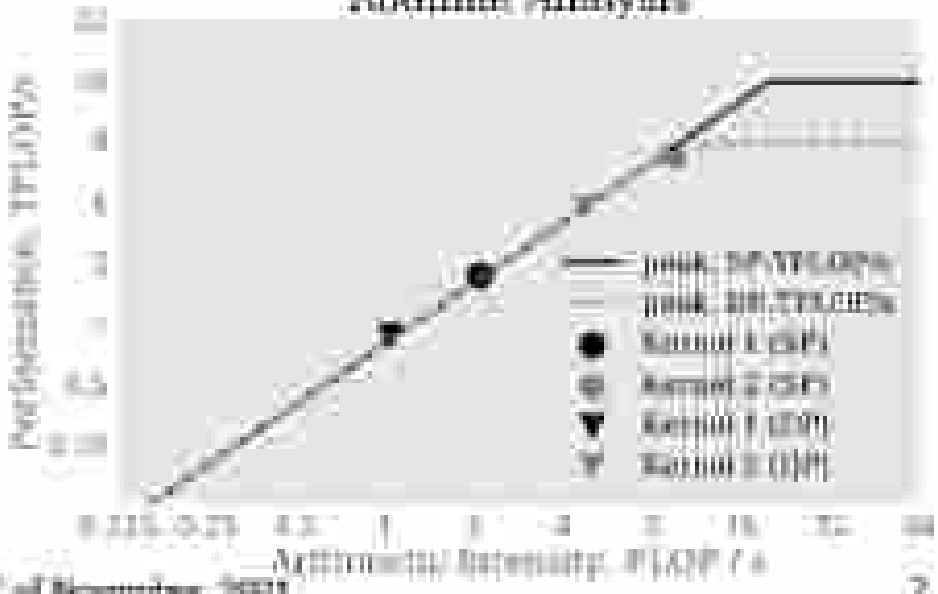
Benchmark:

$$L_e = D \cdot A_e \cdot B_e + L_e \tag{4}$$

where $L_e A \in \mathbb{R}^{n \times 9}$ and $B \in \mathbb{R}^{9 \times 9}$, $D \in \mathbb{R}^{n \times n}$ represents either a mass or stiffness matrix.

Implementation:

$$T_e = A_e \cdot B_e$$
$$L_e = D \cdot T_e + L_e \tag{5}$$



Weak Scaling on Marconi 100

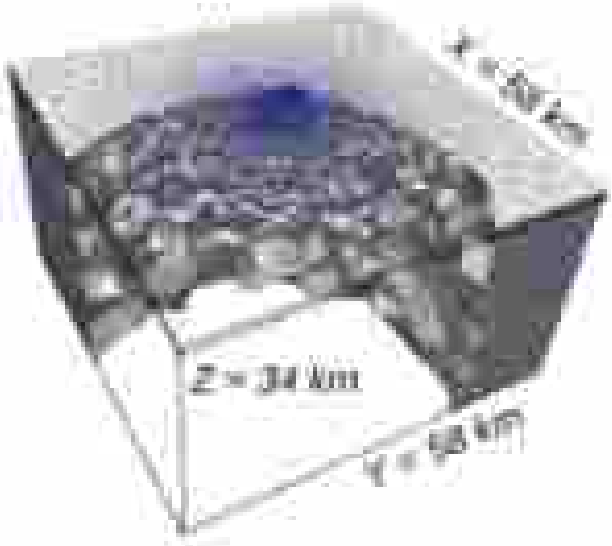
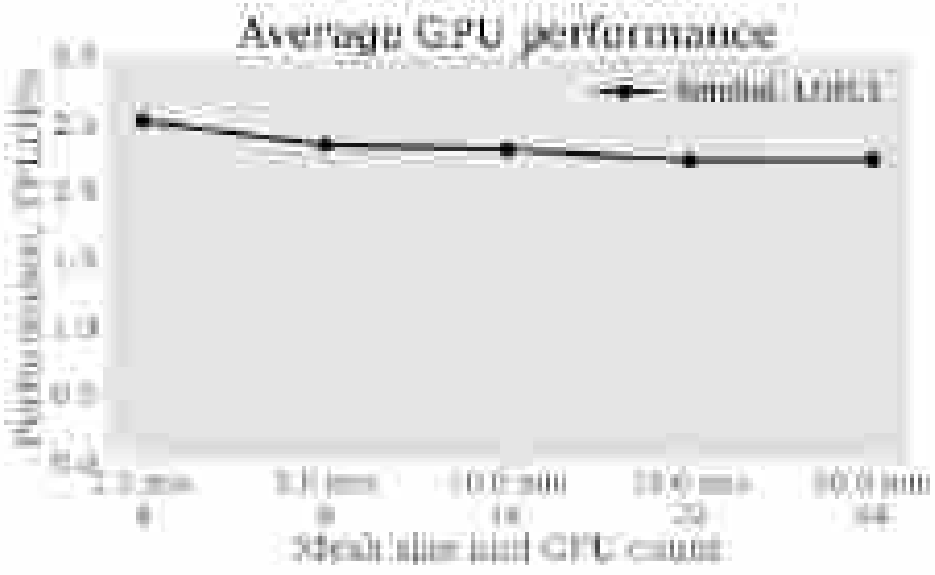


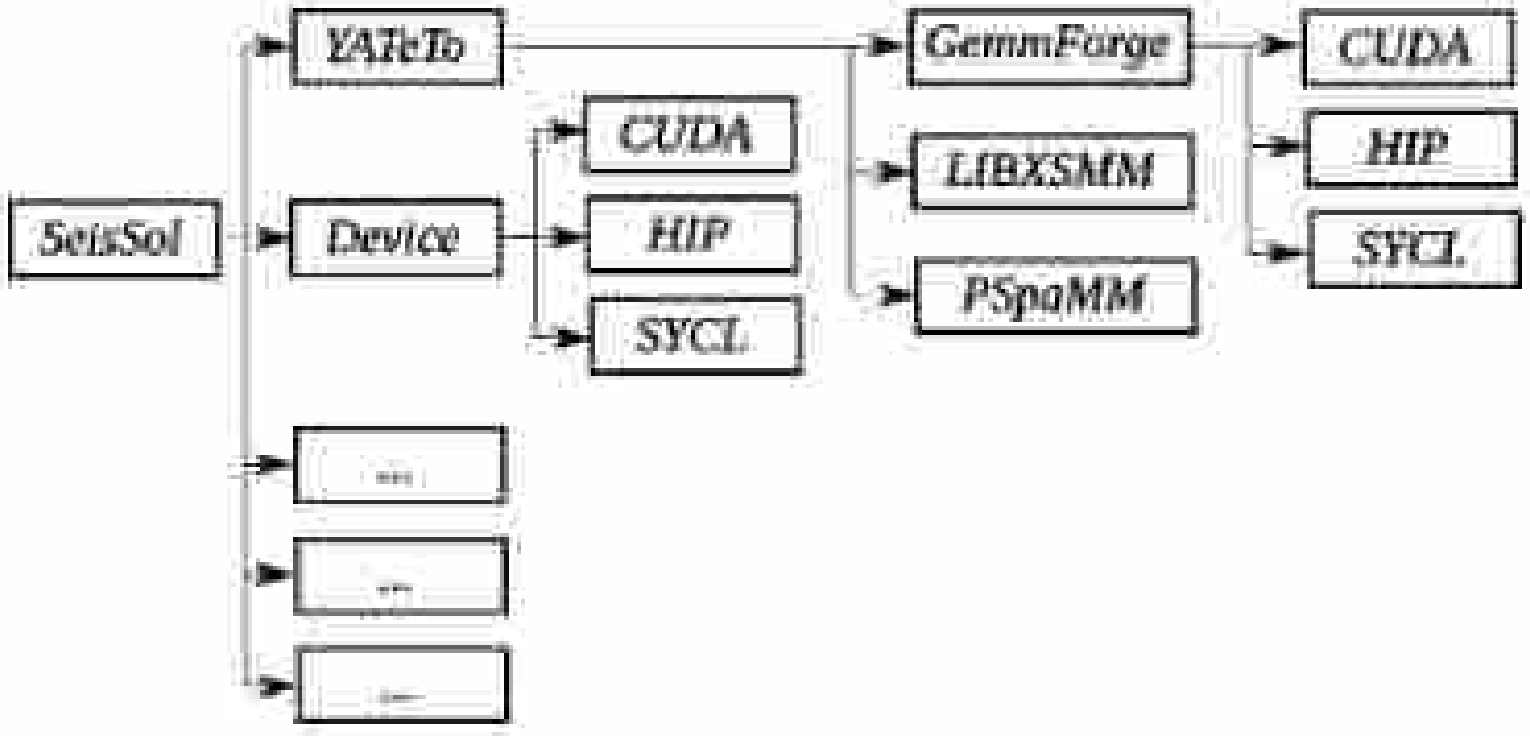
Figure: Layer Over Half space (LOH)

- Parameterized CAD model
- Single point source
- 2 layers with different materials



But SeisSol has some problems with Strong Scaling on GPUs

SeisSol Structure

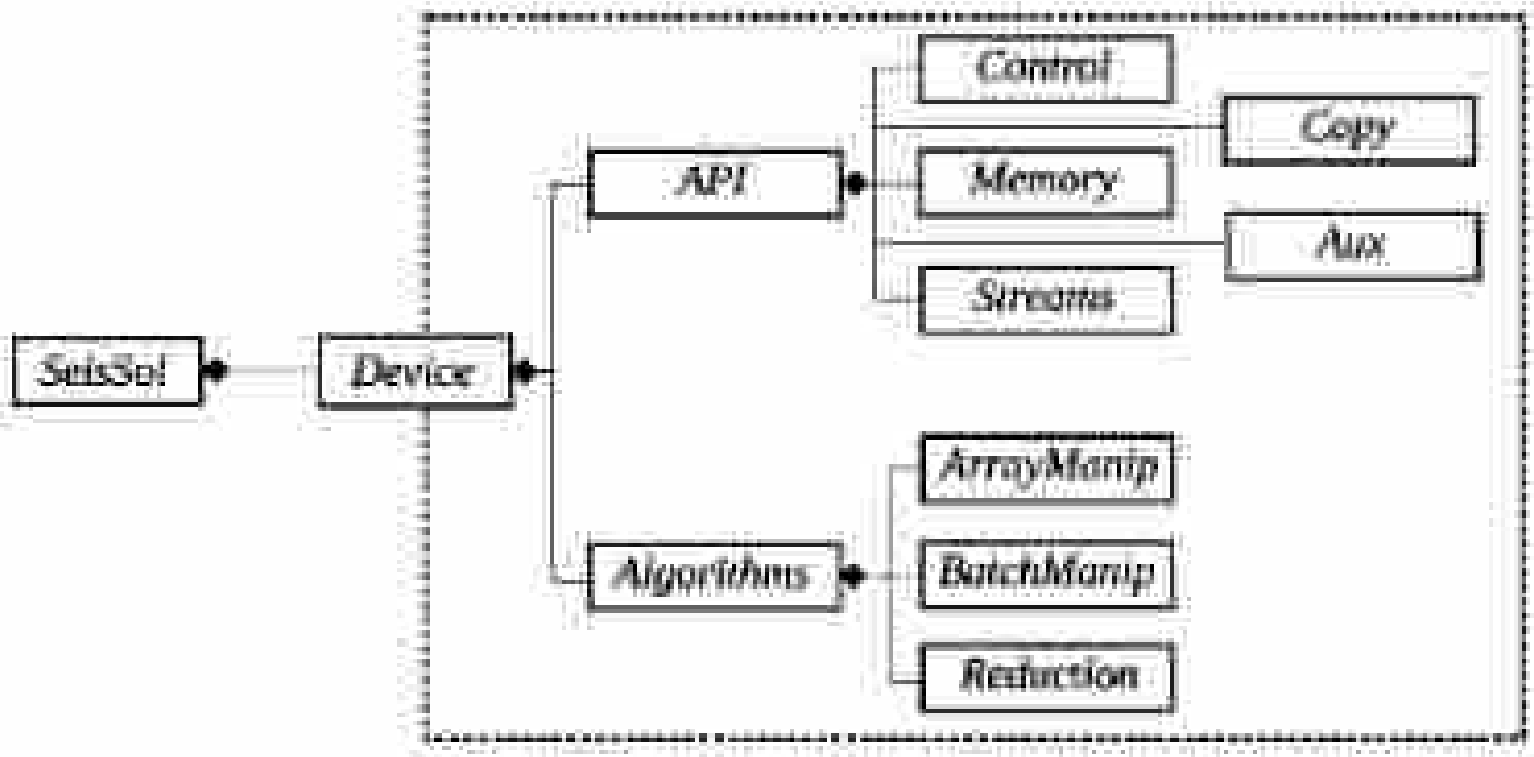


SYCL is a subset of DPC++

- Open language from the Khronos group

Device

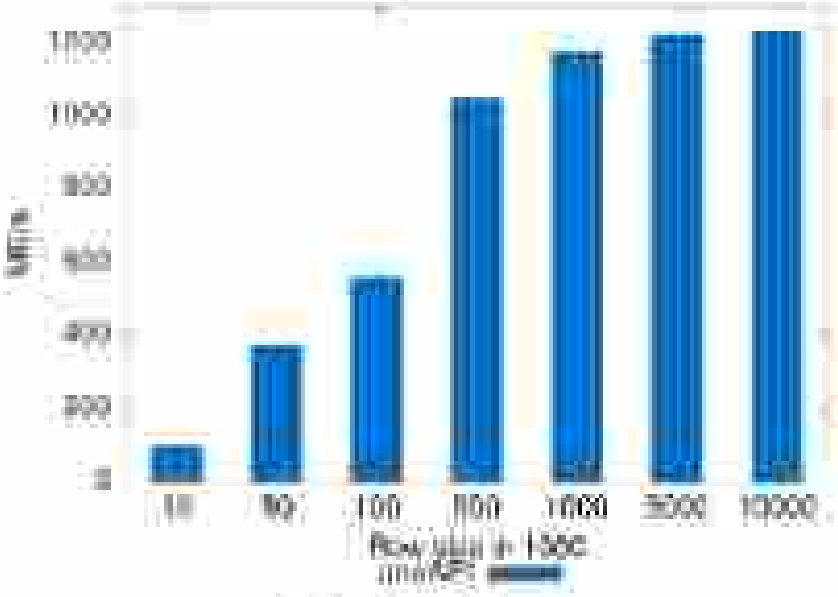
- Provides a simplified interface to GPUs
- Adapts specific GPU APIs and algorithms
- Separate compilation of CPU and GPU source codes
  - ! No GPU specific data types in SeisSol



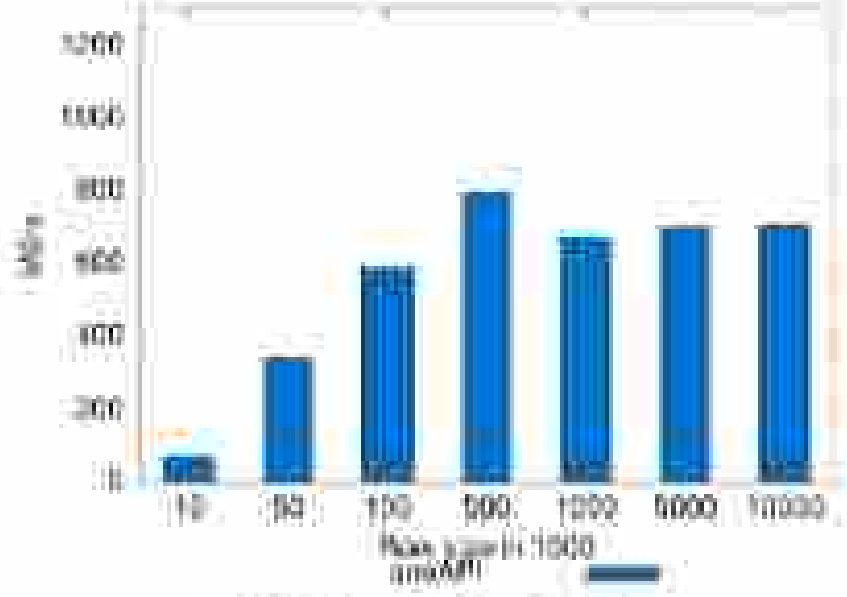
Parallel Jacobi Benchmark (PJB)

- Approximation of a system of linear equations $Ax = b$
- Fix point iteration algorithm
- Parallel implementations GEMV and AXPY on GPUs
- Uses collectives of GPU-Aware MPI
- Uses Device API i.e., *Control*, *Memory* and *Copy* components
- Measures throughput in MB/s for Computations and Communication

Computational Throughput



Iris Xe Max

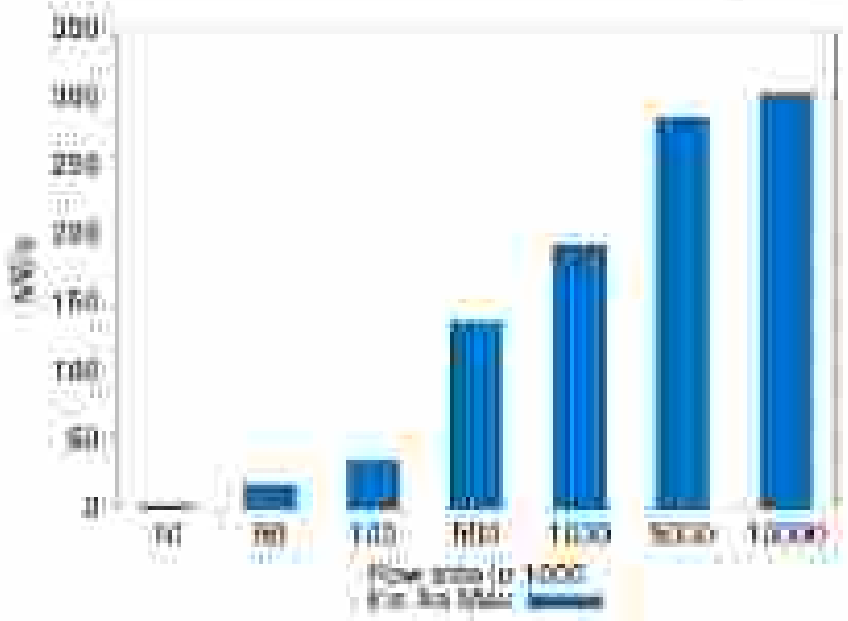


UHD Graphics P630

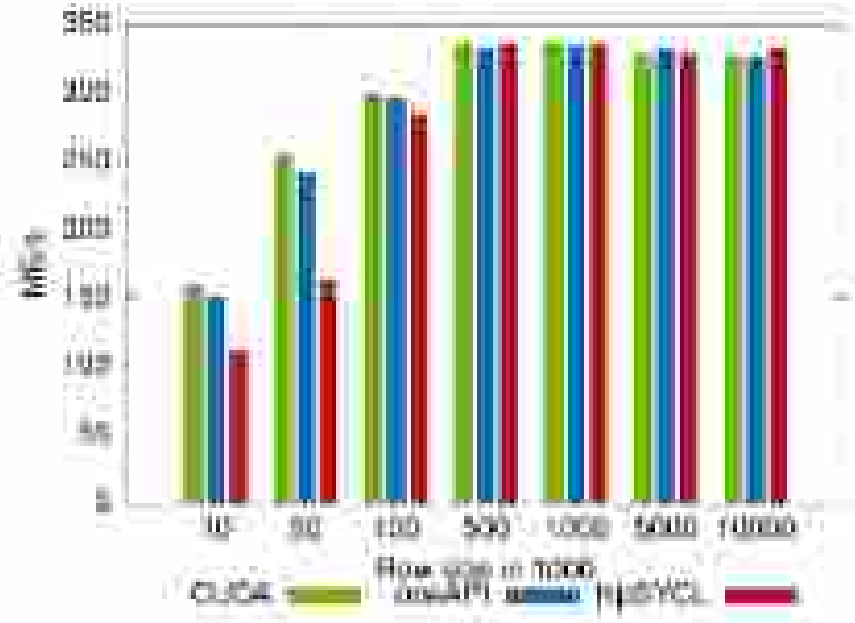
PJB — Communicational Throughput

Intel MPI vs. OpenMPI with 2 GPUs

- Performance of Intel GPU-Aware MPI is not going affect SeisSol
- SeisSol relies on non-blocking P2P communication



Iris Xe Max - Intel MPI



RTX3090 - OpenMPI

GemmForge

$$T_e = A_e \cdot B_e \tag{6}$$

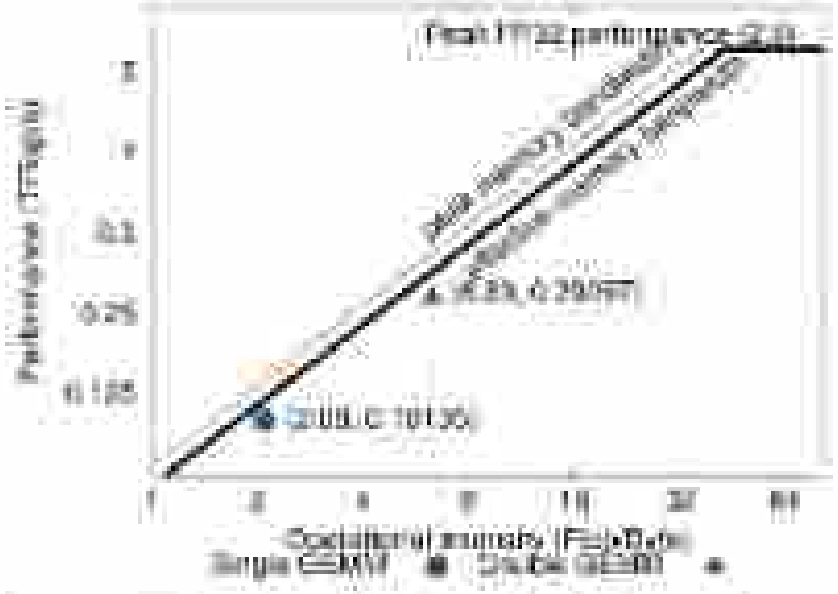
$$L_e = D \cdot T_e + L_e \tag{7}$$

where $L_e, A_e \in \mathbb{R}^{56 \times 8}$, $B_e \in \mathbb{R}^{8 \times 9}$, $D \in \mathbb{R}^{56 \times 56}$

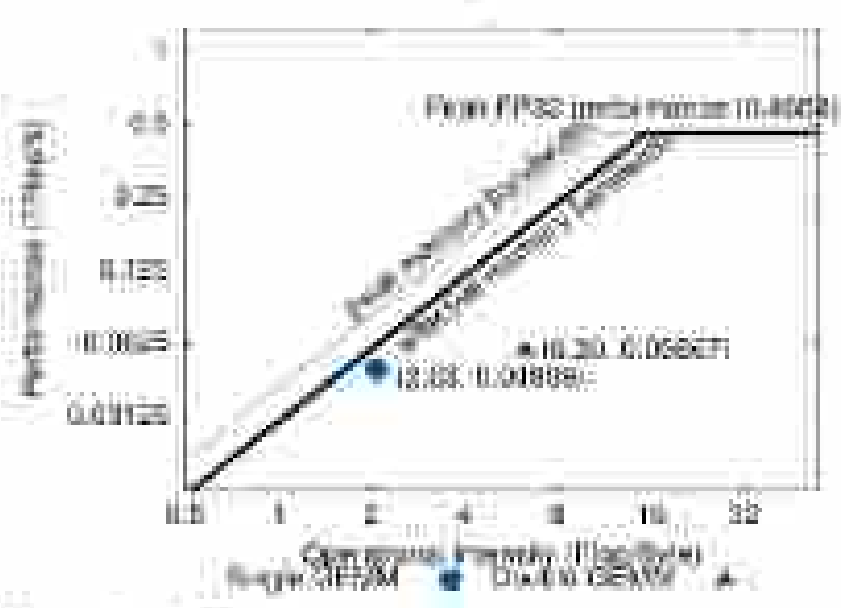
GemmForge Benchmark specification:

- Single GEMM: Equation 6
- Double GEMM: Equations 6 and 7

Intel Iris Xe

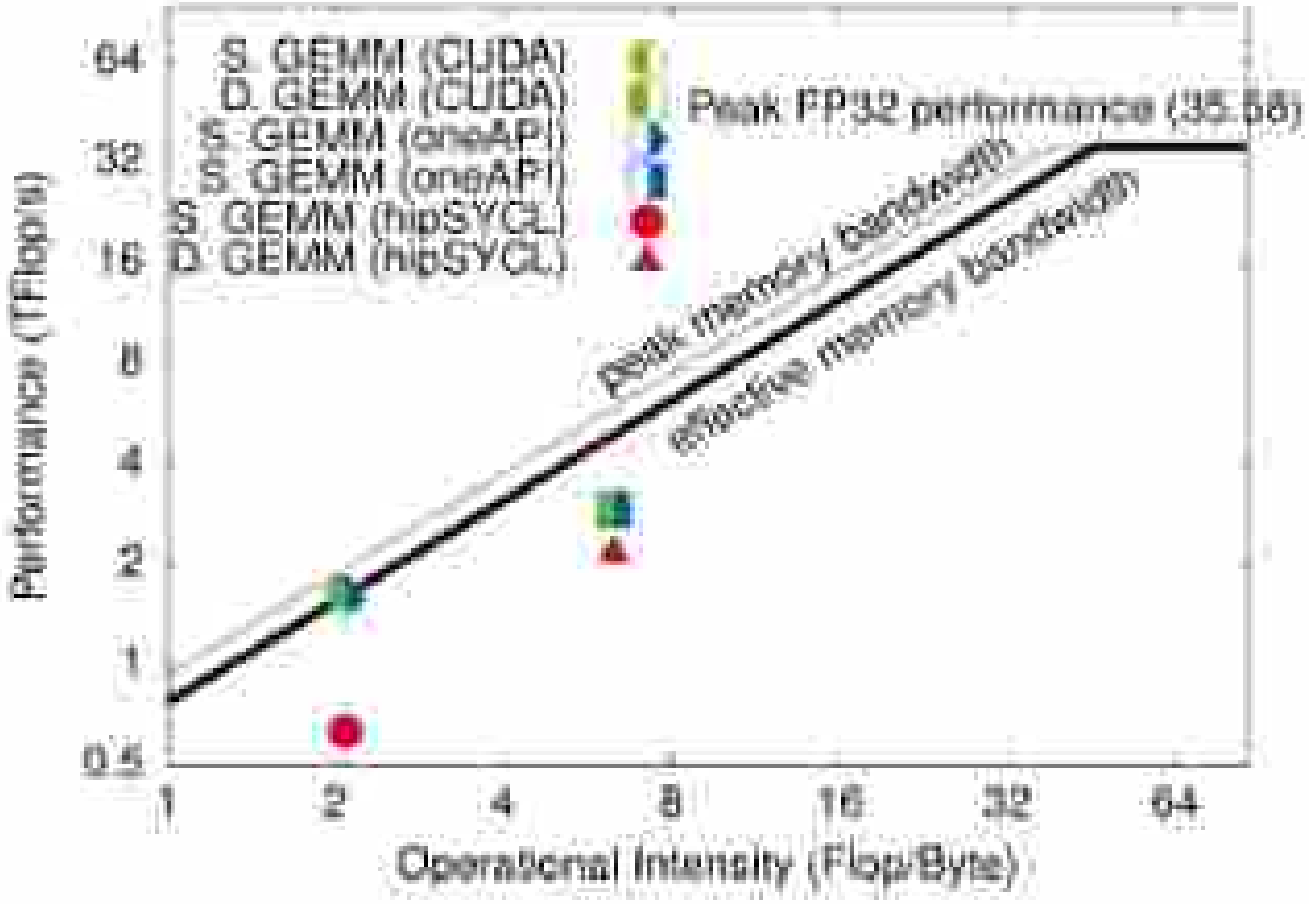


Intel UHD Graphics P630



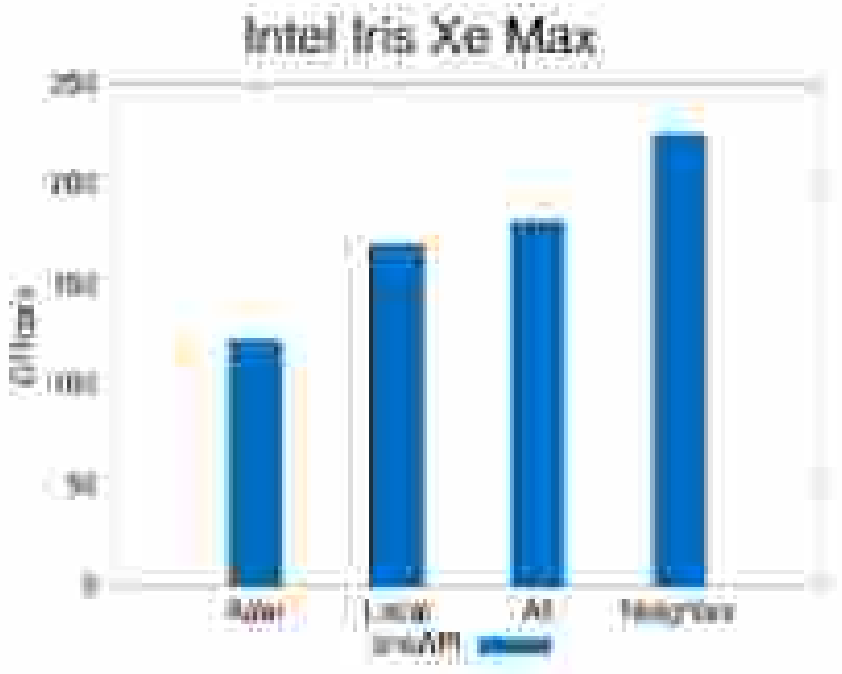
GemmForge

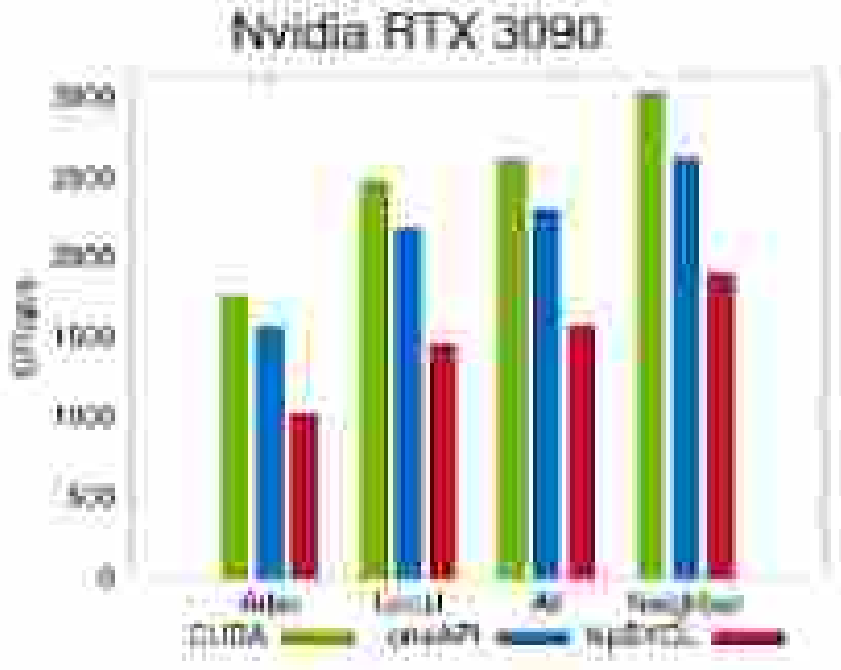
Nvidia RTX 3090 Turbo



SeisSol Proxy

- Contains main computational macro-kernels
- Based on a single time cluster
- Allows to test:
  - Individual macro-kernels
  - Combinations of them





Simulations: LOH.1

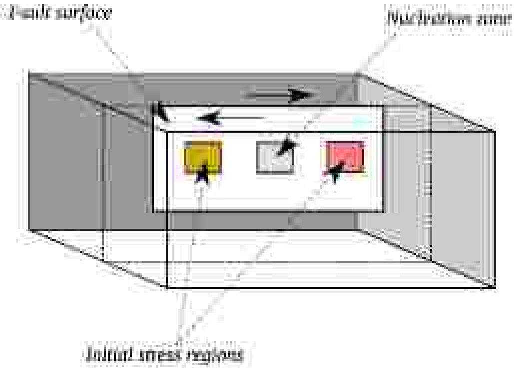
- 2 RTX3090 Turbo GPUs using in a single node
- Using CUDA-Aware OpenMPI via UCX
- Single MPI process per GPU
- Using a dedicated communication thread per MPI process
- Compiled with DPCPP
- Mesh with 1 million elements

Achieved:

- Correct numerical results
- Expected performance based on benchmark results

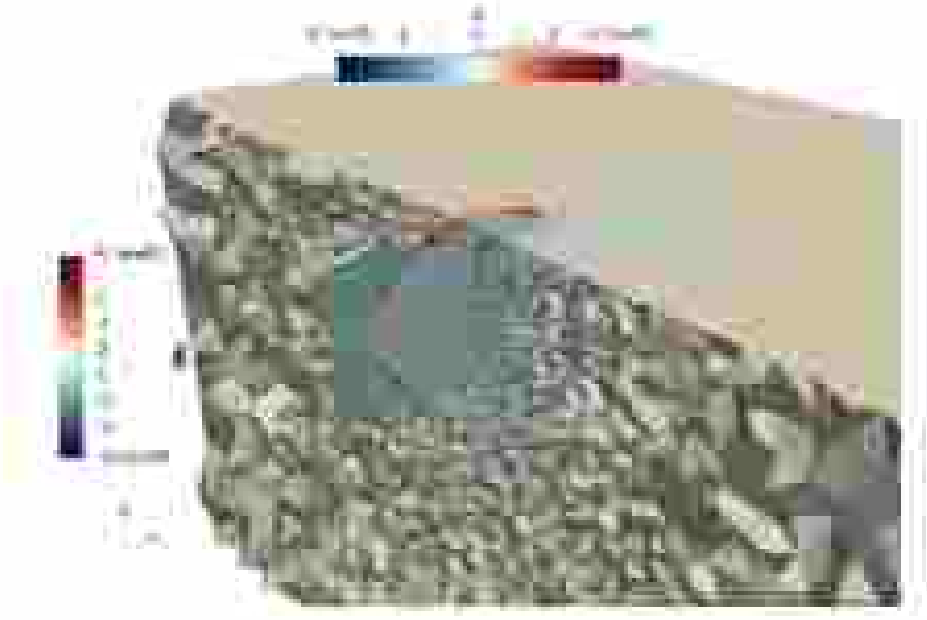
Simulations: TPV5

- Vertical strike-slip fault in homogeneous half-space
- Rupture is artificially nucleated in a square zone at the center
- Two square patches with initial stress conditions
  - higher on the left
  - lower on the right



Simulations: TPV5

- Using the same hardware setup as for LOH.1
- Computing Dynamic Rupture on CPU using all available cores
- Mesh with 2 million elements



Achieved:

- Correct numerical results
- Expected performance

SeisSol CI

- Added SYCL to *SeisSol Device*, *GemmForge* CI pipelines
- Testing SYCL with hipSYCL
- Docker containerization for reproducibility
- CI runner equipped with a single Nvidia TITAN Xp

Conclusion & Future Work

Conclusion:

- Managed into integrate SYCL model into SeisSol
- Correct numerical results
- Expected performance on RTX 3090
- Worked with only 2 isolated components i.e, *Device*, *GemmForge*
- SYCL may replace *Device* in a long-term future

Future work:

- Merging to SeisSol's master branch
- Improving SeisSol building system
- Integrating SYCL into *ChainForge* fused-GEMM generator
- Implementing static SeisSol GPU code with OpenMP offloadings
  - Plasticity model
  - Dynamic rupture

References I

[1] Ravil Dorozhinskii and Michael Bader. "SeisSol on Distributed Multi-GPU Systems: CUDA Code Generation for the Modal Discontinuous Galerkin Method". In: *The International Conference on High Performance Computing in Asia-Pacific Region*. 2021, pp. 69–82.

[2] Kratul Ludwig. "Performance Portability and Evaluation of Heterogeneous Components of SeisSol Targeted to Upcoming Intel HPC GPUs". Bachelorarbeit. Technical University of Munich, 2021.

[3] Carsten Uphoff and Michael Bader. "Yet Another Tensor Toolbox for discontinuous Galerkin methods and other applications". en. In: *ACM Transactions on Mathematical Software* 46.4 (2020). DOI: 10.1145/3406835.