SysGenX Project: Porting to GPU on LUMI-G

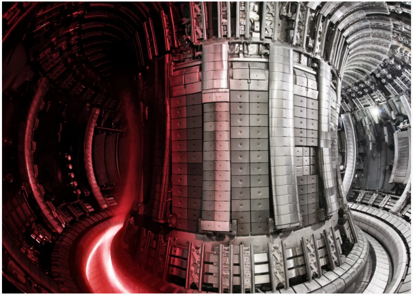
p-multigrid with Dolfinx on GPUs

Igor A. Baratta

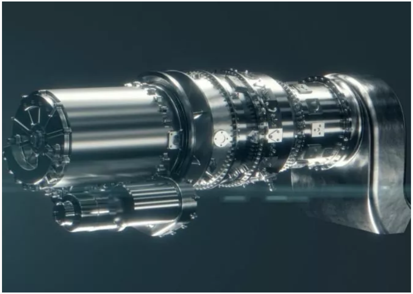
December 7, 2023

Department of Engineering

Background - SysGenX



Nuclear Fusion



Turbo-generator technology

Goal

Efficiently solve the problem

$$A_h x = b \tag{1}$$

using an **iterative** approach:

$$x_k = x_{k-1} + c_k \ .$$

Requirements:

- computing c_k should be ”cheap” and ”scalable”
- the correction should be good enough c_k
- use DOLFINx functions as building blocks.

Model Problem - Code Generation

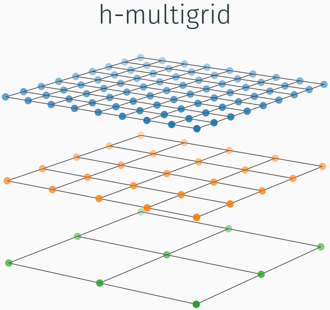
Find $u \in V$ such that:

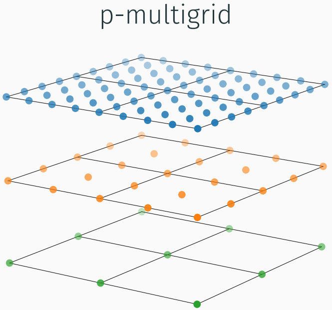
$$\int_\Omega \nabla u \cdot \nabla v \, dx = \int_\Omega f v \, dx \tag{2}$$

```python
V = fem.functionspace(mesh, ("Lagrange", degree))

u = ufl.TrialFunction(V),
v = ufl.TestFunction(V)

# Classical bilinear form
a = ufl.inner(ufl.grad(u), ufl.grad(v)) * ufl.dx
a0 = fem.form(a, dtype=dtype)
```

We're more interested in the case where degree $P \geq 3$.

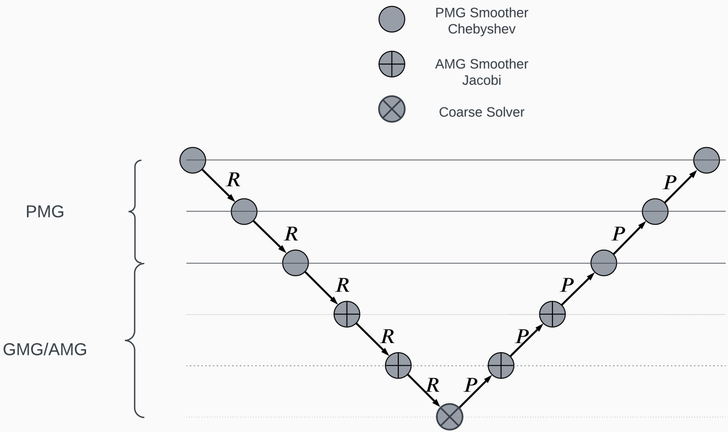
Multigrid Operators





1. Apply ν_1 pre-smoothing steps (G) - $x_k = Gx_{k-1}$
2. Compute residual - $r_h = A_h x_k - b$
3. Restrict residual - $r_H = R r_h$
4. Solve coarse problem - $e_H = A_H^{-1} r_H$
5. Prolong correction - $e_h = P e_H$
6. Correct- $x_k = x_k + e_h$
7. Apply post-smoother (G) - $x_k = Gx_{k-1}$

PMG Implementation

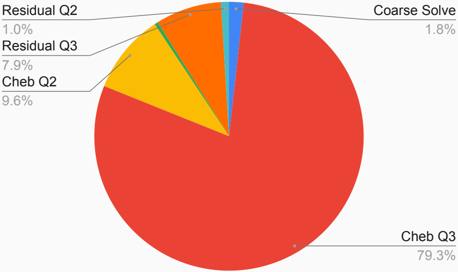


Timing - ”Serial”

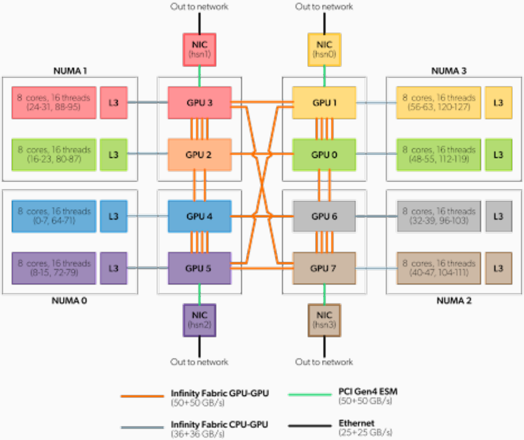
Almost 90% of the time on the fine grid.

Hierarchy:

- Level Q3: dofs 4,999,696 - nnz 616’230’976
- Level Q2: dofs 1,494,425 - nnz 93,773,201
- Level Q1: dofs 191,748 - nnz 4,999,696



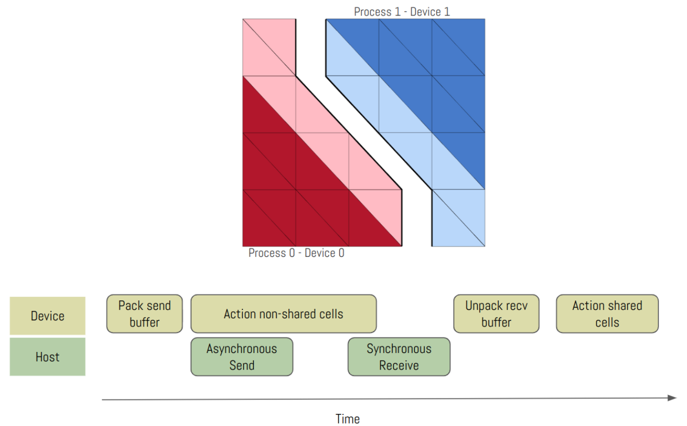
GPU nodes - LUMI-G



2978 nodes with 4 AMD MI250x GPUs and a single 64 cores AMD EPYC ”Trento” CPU
Programming model: MPI + (HIP / SYCL)

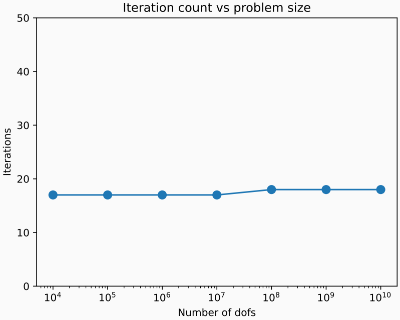
Parallelization

- Vector updates, such as error correction, are performed locally.
- Linear Operators: can be computed separately for ”shared” and ”non-shared” cells.

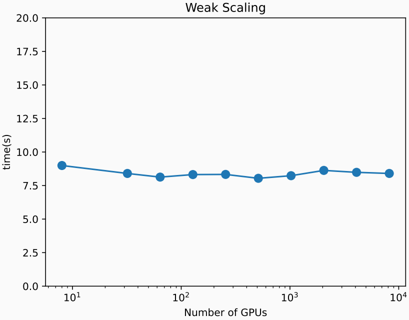


Scaling Results

- Tests on LUMI supercomputer:
  - 3rd place position on the TOP 500 list (1st place in Europe)
- Up to 8192 devices (40 % of full supercomputer).
- Up to 100 billion dofs.

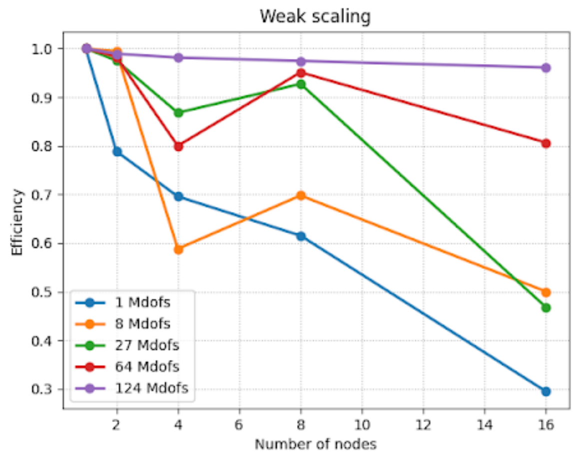


Iteration count remains ”constant” as we increase the problem size.



Nearly constant time-to-solution as the problem size is proportionally increased.

Strong Scaling



Conclusion

- Dolfinx provides the necessary building blocks for implementing PMG and GMG.
- The implementation of these methods can be optimized for parallel execution by overlapping computation and communication.
- In the strong scale limit both intra-node performance and scaling suffer:
  - Latency is a major issue
  - Difficulty to hide communication (<30% parallel efficiency)