Proxy-FUn status and roadmap

https://github.com/numpex/proxy-FUn/

https://gitlab.inria.fr/numpex-pc5/wp2-co-design/proxy-FUn

Proxy-FUn: identifies different motifs from the demonstrator FUnTiDES and generates enhanced, optimized variants of these motifs for reintegration into the demonstrator.

FUnTiDES

- addresses PDEs of the form $\mathbf{M}\frac{d^2\mathbf{p}}{dt^2} + \mathbf{K}\mathbf{p} = \mathbf{f}$
- Utilizes unstructured spectral finite element approximation along with Gauss-Lobatto quadrature points
- implemented in C++20,
- employs Kokkos abstraction to enable numerical kernels to run on various compute units (CPU, GPU)

Explore:

- Different spectral finite element libraries and frameworks.
- Assess the performance of multiple small dense tensor contraction libraries, such as MAGMA, Feel++, and others.
- Analyze how data transfer affects the performance of various implementation strategies.
- Review synchronization techniques, including atomic operations and coloring.
- Investigate optimization methods, like modifying execution policies in Kokkos.

Proxy-kernels:

- Focus on bake-off kernels of the form

$$\mathbf{K} = c^2 \sum_{p,q,r} w_p w_q w_r |\mathbf{J}_{pqr}| \sum_{\alpha=1}^{3} \sum_{\beta=1}^{3} G^{\alpha\beta}_{pqr} \mathcal{D}^{\alpha} \otimes \mathcal{D}^{\beta} \qquad (\mathbf{Ku})_{ijk} = c^2 \sum_{\alpha,\beta=1}^{3} \mathbf{D}^T_{\alpha} \left[\mathbf{W} \odot |\mathbf{J}| \odot \mathbf{G}^{\alpha\beta} \odot (\mathbf{D}_{\beta}\mathbf{u}) \right]$$

- Explore:
  - Various finite/spectral element libraries and frameworks.
  - Benchmark various small dense tensor contraction libraries. (MAGMA, feel++,,...)

- Existing proxy-kernel implementations consist of:
  - Makutu implementation: exhibits $O(n^6)$ complexity, featuring dynamically computed Jacobians optimized for GPUs
  - Tensorial implementation: operates with $O(n^4)$ complexity
  - GEMM-based implementation: designed to leverage BLAS libraries and accelerator backends (cf. Alexandre Roger presentation)

Current strategy implementations in proxy-FUn (acoustic case)

	MAKUTU	TENSORIAL	GEMM	TEAM_VECTOR_GEMM
Outer parallelism	RangePolicy (elements)	RangePolicy (elements)	RangePolicy (elements)	TeamPolicy (elements)
Inner parallelism	None	None	None	TeamVectorRange (nodes/GEMM)
Best for	Reference / low order	Low-to-mid order	CPU / mid-to-high order	GPU / high order

Proxy evolutions:

Proxy-FUn:

- Customize proxy-Fun version to handle various motifs:
  - Communication:
    - at the node level and compute node level, including multi-GPU setups
    - StarPu,
    - Domain decomposition,…
  - I/O operations…

Proxy-Kernels:

- Use feel++ to approximate the different integrals operators defined in proxy-FUn to approximate WE solution
- Continue to explore kernels implementation strategies

Benefits: include a more tailored approach, increased ease of use, and improved sharing capabilities with other working groups. But requires standard API connecting proxy-Kernels to proxy-FUn and FUnTiDES.

Metric definitions to assess and compare performances:

- Roofline,
- Profiling,
- Any tools coming from NumPEx WG?