Accelerating an high-order mesh optimiser using an architecture-independent programming model

Imperial College London

Jan Eichstädt, David Moxey, Mashy Green, Michael Turner, and Joaquim Peiró

Motivation

Bottlenecks towards the adoption of high-order methods in industry are the availability of **robust meshing capabilities**, and their **efficiency on heterogeneous HPC systems**.

High-Order Mesh Generation

High-order meshes are generated by deforming linear meshes to conform to the curved CAD geometry and a subsequent element distortion using an elastic body analogy to **improve the mesh quality**.

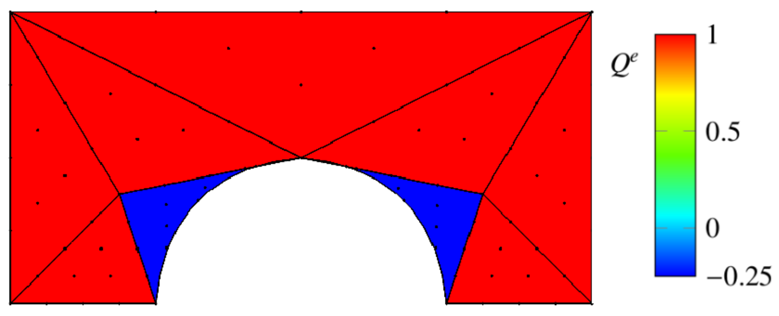

Figure 1: Initial straight-sided but boundary-conforming mesh

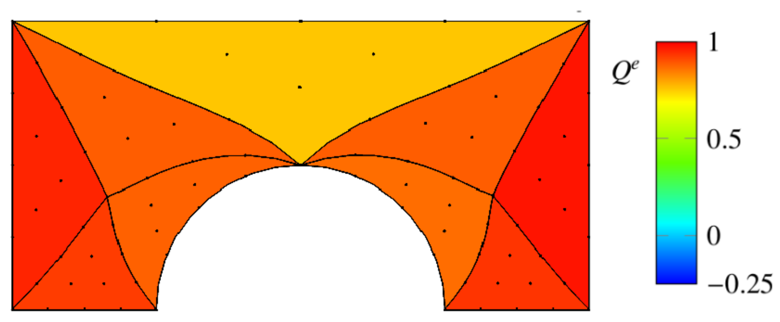

Figure 2: Curvilinear mesh after hyper-elastic optimisation

Parallel Mesh Optimisation Method

The optimisation is implemented within the meshing tool *NekMesh* [1], which is part of the spectral/*hp* element suite *Nektar++* [2], using a variational framework [3].

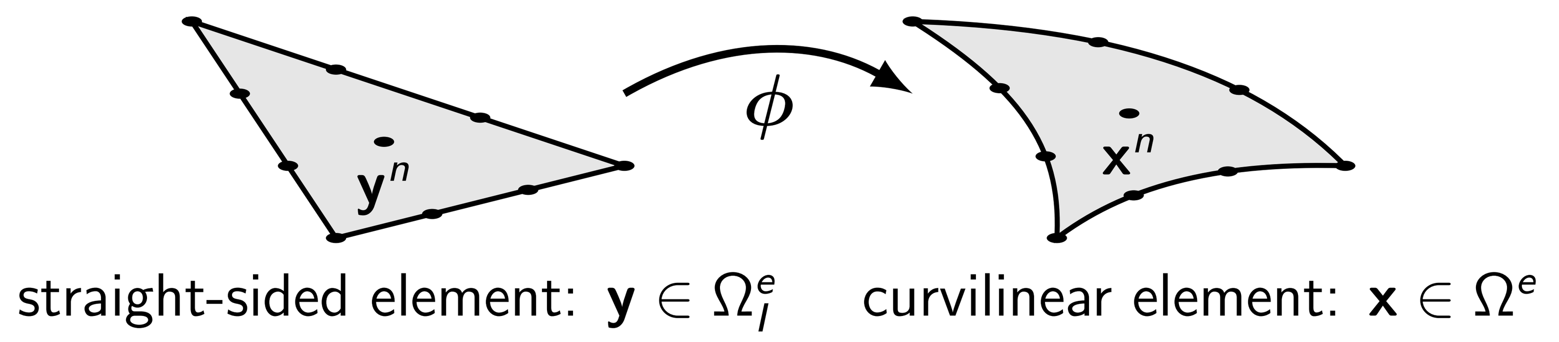

Figure 3: Mapping between straight-sided, and curvilinear element

The mesh is processed by minimising an energy functional $\mathcal{E}$, that is a function of the mesh deformation ϕ.

$$\text{find } \min_{\phi} \mathcal{E}(\nabla\phi) = \int_{\Omega_I} W(\nabla\phi)\, \mathrm{d}\mathbf{y} \tag{1}$$

The global minimisation is solved with a relaxation approach, to calculate the energy functional based on the element subset of node i. This allows the **parallel processing of independent mesh nodes**.

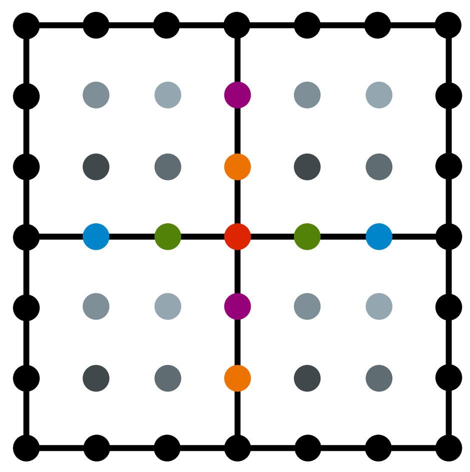

Figure 4: Example domain; nodes with the same colour are processed in parallel.

$$\text{find } \min_{\phi} \mathcal{E}_i(\nabla\phi) = \sum_{e \subset i} \int_{\Omega_I^e} W(\nabla\phi)\, \mathrm{d}\mathbf{y} \tag{2}$$

The coordinates $\mathbf{x}$ of node i are then updated using a Newton method.

$$\mathbf{x}_i^{k+1} = \mathbf{x}_i^k - \alpha \mathbf{H}(\mathcal{E}_i)^{-1}\mathbf{G}(\mathcal{E}_i) \tag{3}$$

References

[1] M. Turner, D. Moxey, S. J. Sherwin, and J. Peiró. Automatic Generation of 3D Unstructured High-Order Curvilinear Meshes. In: *ECCOMAS Congress 2016 VII European Congress on Computational Methods in Applied Sciences and Engineering*. 2016, pp. 5–10.

[2] C. D. Cantwell, D. Moxey, A. Comerford, A. Bolis, G. Rocco, G. Mengaldo, D. De Grazia, S. Yakovlev, J. E. Lombard, D. Ekelschot, B. Jordi, H. Xu, Y. Mohamied, C. Eskilsson, B. Nelson, P. Vos, C. Biotto, R. M. Kirby, and S. J. Sherwin. Nektar++: An open-source spectral/hp element framework. *Computer Physics Communications* 192 (2015), pp. 205–219. DOI: 10.1016/j.cpc.2015.02.008.

[3] M. Turner, J. Peiró, and D. Moxey. A Variational Framework for High-Order Mesh Generation. *Procedia Engineering* 163 (2016), pp. 340–352. DOI: 10.1016/j.proeng.2016.11.069.

[4] H. Carter Edwards, C. R. Trott, and D. Sunderland. Kokkos: Enabling manycore performance portability through polymorphic memory access patterns. *Journal of Parallel and Distributed Computing* 74.12 (2014), pp. 3202–3216. DOI: 10.1016/j.jpdc.2014.07.003.

Kokkos Programming Model

We employ the *Kokkos* library [4] to express the parallelism with **architecture-independent abstraction layers** within a single code-base, that is compiled with different back-ends such as *OpenMP* or *CUDA* to work efficienctly on different architectures.

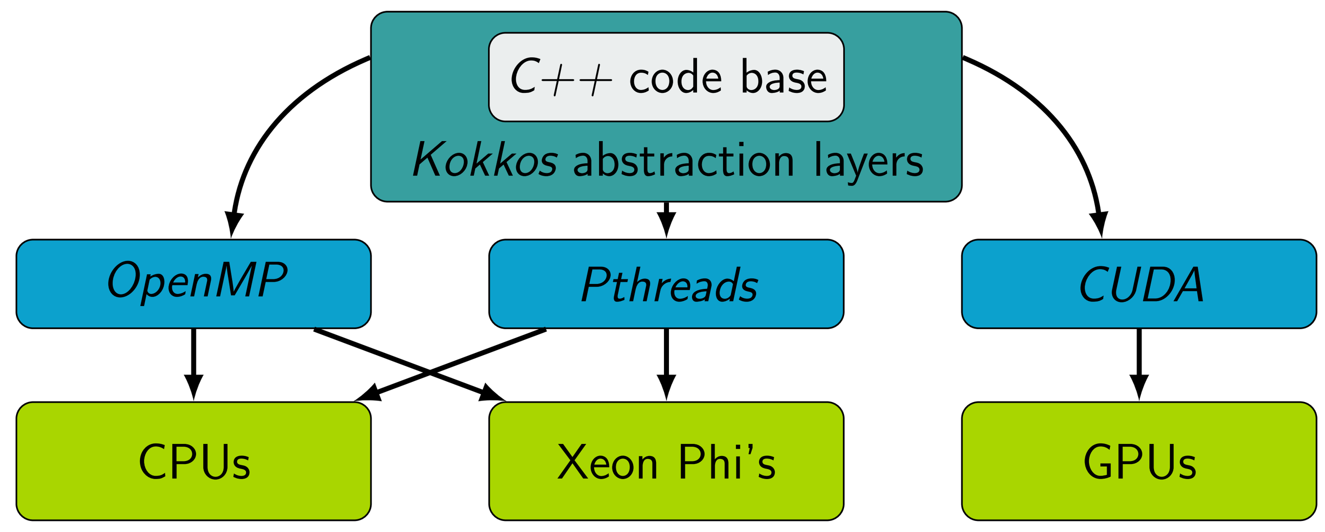

Figure 5: Implementing the *Kokkos* programming model

Performance Results

■ vs ●: The full *Kokkos* version is 9.5-times faster than the native implementation and can benefit from hyper-threading.
▼ vs ●: *Kokkos* data-structures lead to superior performance.

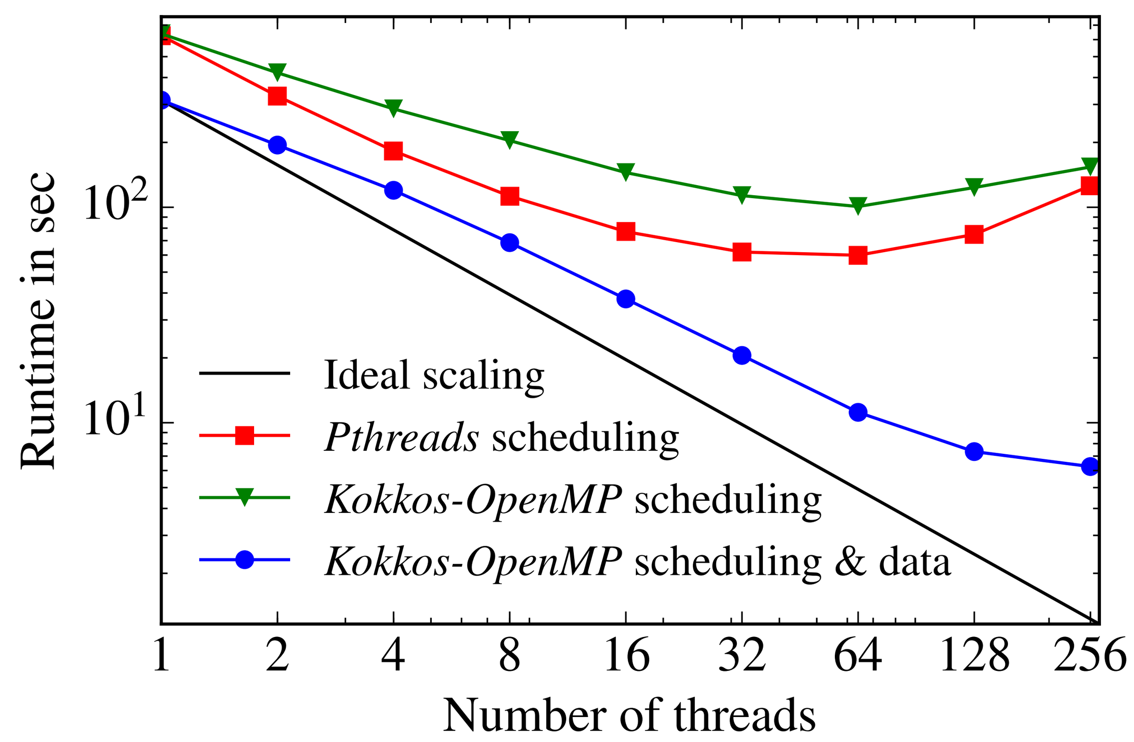

Figure 6: Strong scaling of 3rd-order tetrahedral case on Xeon Phi 7120 (Knights Landing) accelerator using up to 256 threads on 64 cores.

● ● vs ● ●: The GPUs outperform both multi-core systems and are less dependent on mesh polynomial orders.

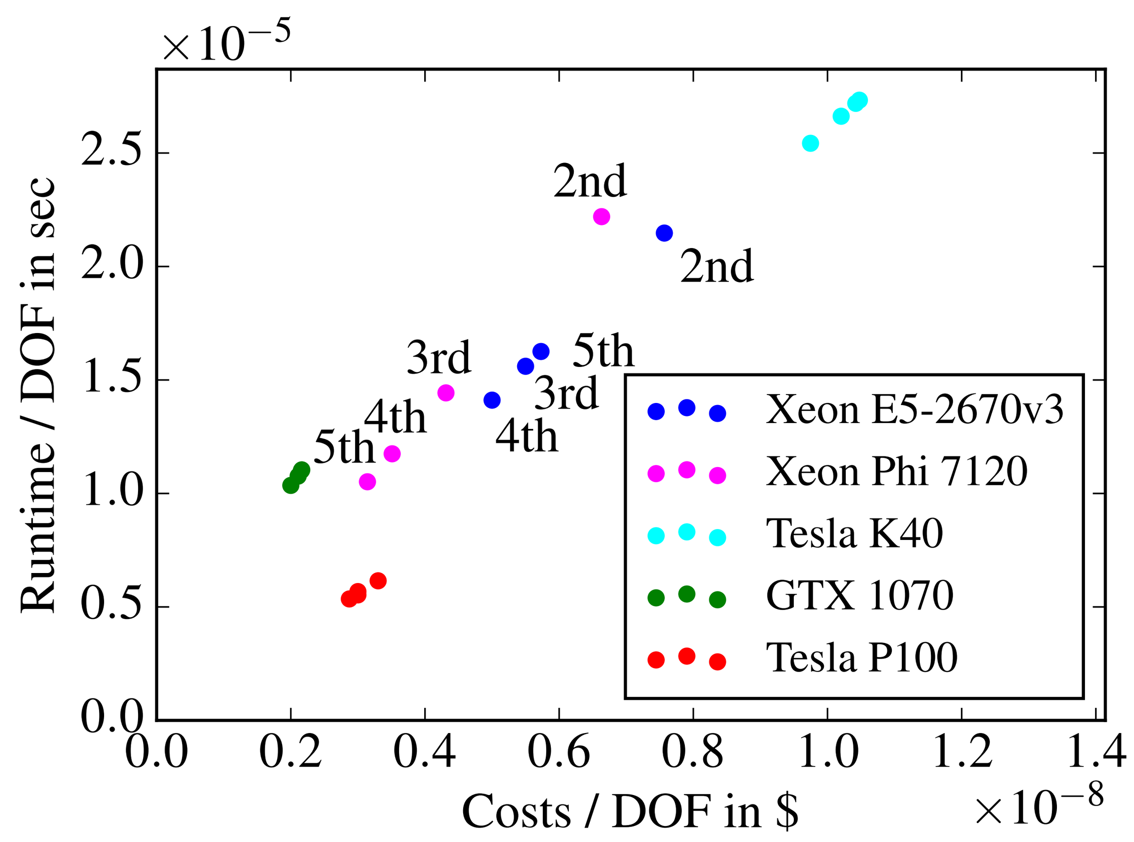

Figure 7: Costs (*runtime × monthly prices of equivalent bare-metal cloud computing systems*) vs runtime for tetrahedral meshes of different orders.

Summary

We have implemented a high-order mesh optimiser using *Kokkos*, an architecture-independent programming model. We achieve better performance than a native multi-core implementation on CPUs and Xeon Phi's. Without changing the code-base we can realise even further cost and time savings by leveraging the *Kokkos* portability to GPU architectures using the *CUDA* backend.

J.E. gratefully acknowledges the support by the President's Scholarship of Imperial College London.