Improving Strong Scalability Limits of Finite Element Based Solvers

Niclas Jansson
KTH Royal Institute of Technology
Stockholm, Sweden
njansson@kth.se

ABSTRACT

Current finite element codes scale reasonably well as long as each core has sufficient amount of local work that can balance communication costs. However, achieving efficient performance at exascale will require unreasonable large problem sizes, in particular for low-order methods, where the small amount of work per element already is a limiting factor on current post petascale machines. One of the key bottlenecks for these methods is sparse matrix assembly, where communication latency starts to limit performance as the number of cores increases. We present our work on improving strong scalability limits of message passing based general low-order finite element based solvers. Using lightweight one-sided communication, we demonstrate that the scalability of performance critical, latency sensitive kernels can achieve almost an order of magnitude better scalability. We introduce a new hybrid MPI/PGAS implementation of the open source general finite element framework FEniCS, replacing the linear algebra backend with a new library written in UPC. A detailed description of the implementation and the hybrid interface to FEniCS is given, and we present a detailed performance study of the hybrid implementation on Cray XC40 machines.

CCS CONCEPTS

• **Mathematics of computing** → Mathematical software performance; **Solvers**; • **Computing methodologies** → *Massively parallel algorithms.*

KEYWORDS

Sparse Matrices, FEM, PGAS, UPC, Hybrid MPI/PGAS

1 INTRODUCTION

To strongly scale finite element codes on future exascale systems will require either unreasonable large problem sizes to balance communication costs, or algorithmic developments need to be achieved which retain strong scalability even for very few elements per core. In particular for low-order methods, where the small amount of work per element already is a limiting factor on current post-petascale machines. One of the key bottlenecks for these methods is sparse matrix assembly, where communication costs starts to limit performance as the number of cores increases.

The communication costs in matrix assembly come from the underlying discretisation. Unstructured meshes are excellent for accurate approximation of complex geometries. However, the lack of underlying structure implies an unstructured communication pattern. This together with the low amount of work per element of low-order methods can have a negative effect on the overall performance, as can be seen in [1]. Matrix assembly is quickly losing strong scalability, but can often be amortised by much more costly linear solvers. However, at higher core counts the cost of assembling the system starts to become the dominating factor.

The reason for this negative behaviour is partly due to the programming model used (message passing) and its two-sided communication abstraction. For a large number of cores, the need to match send and receive messages will unavoidably increase latency and synchronisation costs. Non-blocking communication is often employed to lower this cost, albeit it requires the receiver to occasionally check for messages which introduces latency and additional overhead costs.

To address these issues we use the lightweight, low latency one-sided communication offered by Partitioned Global Address Space (PGAS) languages to build a new sparse matrix representation. This new representation is used to increase sparse matrix assembly in a hybrid MPI/PGAS finite element solver based on the open source framework FEniCS [3], where we focus on replacing key latency dominating parts of the linear algebra backend with the new PGAS based implementation. The main contribution of our work is to demonstrate the applicability of PGAS in terms of performance and present a path forward for legacy MPI codes with the hybrid MPI/PGAS model.

2 SPARSE MATRIX REPRESENTATION

For the new sparse matrix representation we use a Compressed Row Storage (CRS) like format, optimised for random insertion during matrix assembly. The parallelisation is based on a row wise data distribution, where each rank is assigned a contiguous set of rows. We adhere to the two phase assembly process, where entries are inserted into a matrix in two steps, first local entries are inserted and all other entries are placed in a local staging area. In the second step each rank fetches parts of other ranks' staging areas using remote memory copy. The implementation is written in Unified Parallel C (UPC) [5], and uses a directory of objects representation to overcome the fixed block size global memory allocation in UPC. Furthermore, the implementation provides a hybrid interface, with data accessed through an opaque interface allowing for interfacing with non-PGAS based languages.

3 HYBRID PARALLELISATION OF FENICS

FEniCS is a high-level problem-solving environment for automated solution of partial differential equations (PDEs) by the finite element method. The core part of FEniCS is the Object-Oriented finite element library DOLFIN [4], from which we have developed a high performance branch [2] for distributed memory architectures. DOLFIN handles mesh representation and finite element assembly but relies on external libraries for solving the linear systems.

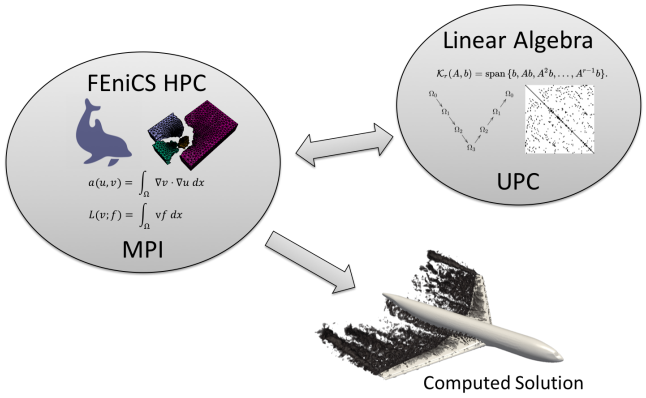

Figure 1: The hybrid MPI/PGAS parallelisation of FEniCS.

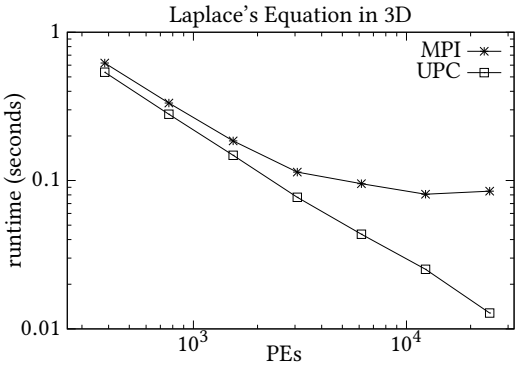

Figure 2: Assembly time (in seconds).

DOLFIN is written in C++, parallelised using MPI, and uses ParMETIS for mesh partitioning and PETSc for linear algebra. In this work we simply change the linear algebra backend to our new UPC based implementation, and access matrices and vectors from DOLFIN through the opaque interface, as illustrated in Figure 1.

4 PERFORMANCE EVALUATION

To evaluate the feasibility of using hybrid MPI/PGAS methods to improve the scalability limits of matrix assembly in low-order finite element codes we tested our new PGAS based sparse matrix implementation on a set of different matrices from various finite element discretisation of partial differential equations, all with different kinds of communication and computational costs. For all cases, we measure the time to recompute the stiffness matrix, assuming that the sparsity pattern is known a priori.

Experiments with the new hybrid parallelisation were run on two different Cray XC40 machines, first the 3944 node Hornet located at HLRS, and secondly the 1676 node Beskow at PDC. The flat MPI version of FEniCS using PETSc was compared against our new hybrid version with a UPC based linear algebra backend.

In one of these benchmarks we compute the stiffness matrix corresponding to the continuous linear Lagrange finite element discretisation of Laplace's equation. This benchmark corresponds to a worst case scenario, since the stiffness matrix can be computed with minimal work. Hence, this benchmark tests the communication cost more than the insertion rate. For this experiment we use an unstructured tetrahedral mesh of the unit cube consisting of 317M elements.

The results from this benchmark is shown in Figure 2, which clearly shows the advantage of the PGAS approach. Since the matrix can be computed with minimal work the performance of the flat MPI FEniCS flattens out when the number of elements per rank becomes too small, while the less latency affected UPC code continues to perform well.

5 CONCLUSIONS

In this work we investigated the strong scalability limits of finite element based solvers, in particular low-order methods which are mostly communication bound due to the low amount of work per element. Using the partitioned global address space abstraction we have investigated the feasibility of hybrid methods to address these scalability limits, combining traditional message passing with low latency one-sided communication within the same applications, reducing the threshold for adopting new programming models in legacy applications. We demonstrated the applicability of this approach with a hybrid MPI/PGAS implementation of the finite element framework FEniCS, where the sparse matrix representation is replaced with one based on PGAS. The performance was evaluated on two different Cray XC40s, where the hybrid model demonstrates great potential with significant performance improvements of the sparse matrix assembly kernels. Furthermore, the hybrid approach offers an alternative to a complete rewrite of legacy MPI codes when preparing them for future platforms.

6 ACKNOWLEDGMENTS

This work has been supported by the EXCELLERAT project and has been funded by the European Commission's ICT activity of the H2020 Programme under grant agreement number: 823691. The computations were performed on resources provided by the PRACE Research Infrastructure resource Hornet based in Germany at Höchstleistungsrechenzentrum Stuttgart (HLRS) and the Swedish National Infrastructure for Computing (SNIC) at PDC Center for High Performance Computing.

REFERENCES

[1] Johan Hoffman, Johan Jansson, and Niclas Jansson. 2016. FEniCS-HPC: Automated Predictive High-Performance Finite Element Computing with Applications in Aerodynamics. In *Parallel Processing and Applied Mathematics*, Roman Wyrzykowski, Ewa Deelman, Jack Dongarra, Konrad Karczewski, Jacek Kitowski, and Kazimierz Wiatr (Eds.). Springer International Publishing, Cham, 356–365.

[2] Niclas Jansson. 2013. *High Performance Adaptive Finite Element Methods: With Applications in Aerodynamics.* Ph.D. Dissertation. KTH Royal Institute of Technology.

[3] Anders Logg, Kent-Andre Mardal, Garth N. Wells, et al. 2012. *Automated Solution of Differential Equations by the Finite Element Method.* Springer. https://doi.org/10.1007/978-3-642-23099-8

[4] Anders Logg and Garth N. Wells. 2010. DOLFIN: Automated finite element computing. *ACM Trans. Math. Softw.* 37, 2 (2010), 1–28.

[5] UPC Consortium. 2005. *UPC Language Specifications, v1.2.* Technical Report LBNL-59208. Lawrence Berkeley National Lab.