OOKAMI PROJECT APPLICATION

Date: **16/11/2021**

Project Title: **Testing and porting G+Smo to A64FX architecture**

Usage:

☒ Testbed

☐ Production

Principal Investigator: Matthias Möller

University/Company/Institute:	Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, Department of Applied Mathematics, Section Numerical Analysis
Mailing address including country:	Mekelweg 4, 2628 CD Delft, The Netherlands
Phone number:	+31 (0)15 27 89755
Email:	M.Moller@tudelft.nl

Names & Email of initial project users:

Matthias Möller, M.Moller@tudelft.nl
Hugo Verhelst, H.M.Verhelst@tudelft.nl
Angelos Mantzaflaris, Angelos.Mantzaflaris@inria.fr

Usage Description:

G+Smo (https://github.com/gismo/gismo) [1] is an open-source C++ library for Isogeometric Analysis (IGA) [2]. The core idea of IGA is to adopt a unified mathematical approach, namely B-Splines and non-uniform rational B-Splines (NUBRS), for modelling geometries and approximating (initial) boundary value problems. This unified mathematical approach enables the seamless integration of computer-aided geometric design and finite element analysis with as advantage shorter and more interactive product development cycles.

The goal of this project is to port the G+Smo source code to the A64FX architecture. G+Smo is based on the open-source library Eigen (https://eigen.tuxfamily.org) which ships with support for ARM's SVE instructions since release 3.4. A large part of G+Smo's own source code (e.g., the matrix and vector assembly routines and the iterative solvers) is moreover parallelized using OpenMP. A few selected applications are moreover parallelized using MPI. For multiple years, G+Smo's primary development platform used to be x86_64 systems. Since recently G+Smo has been successfully ported to ARM-based systems (Apple Silion M1, Huawei Hi1616) and IBM Power 8 systems.

In the first stage of this project, we will test the G+Smo source code 'as is' on the A64FX platform to obtain a baseline performance model of the underlying Eigen library. The findings will be shared with the Eigen community.

In the second phase, we will identify computational bottlenecks in G+Smo's own source code and specialize the relevant OpenMP implementations for the A64FX architecture. As most computations in G+Smo are memory-bound (especially the matrix and vector assembly routines), we expect a high benefit from using a high memory bandwidth architecture. During this phase, the focus is on achieving high single-node performance.

In the third phase, we will benchmark a few of G+Smo's MPI-enabled applications and port them to the A64FX architecture. The primary focus will be on the parallel-in-time multigrid solvers which rely on the open-source XBraid library (https://github.com/XBraid/xbraid) combined with our in-house developed p-multigrid solvers [3] for the efficient solution of spatial problems. In this application, the XBraid library realizes the MPI-communication and the p-multigrid spatial solvers are parallelized using OpenMP. So far, we have performed weak and strong-scalability tests with up to 2,048 cores of a 128 node Intel Xeon Gold 6130 system. These tests showed nearly perfect strong and weak scalability of the code.

Computational Resources:

Total node hours per year: estimated 5000

Size (nodes) and duration (hours) for a typical batch job: For phase 1 and 2, we only require a single node with only very short runtimes per job (<10min) as the focus is on benchmarking and testing individual components of the G+Smo source code. Phase 3 might require a few large-scale runs to benchmark the strong and weak scalability of selected MPI-applications preferably with all nodes available.

Disk space (home, project, scratch): no home/project/scratch requirements beyond default as no simulation outputs will be stored.

Personnel Resources (assistance in porting/tuning, or training for your users**): n/a**

Required software:

- G+Smo (https://github.com/gismo/gismo)
- CMake, Git, Blas, Lapack, MPI (optional)
- C++14 compiler with OpenMP support (GCC, Clang, Arm HPC are known to work)

If your research is supported by US federal agencies:

Agency: n/a

Grant number(s): n/a

References:

[1] B. Juettler, U. Langer, A. Mantzaflaris, S. Moore, and W. Zulehner: Geometry + Simulation Modules: Implementing Isogeometric Analysis. In: Proc. Appl. Math. Mech. 14(1), pp. 961-962, 2014. DOI: 10.1002/pamm.201410461

[2] T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs: Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Comput. Methods Appl. Mech. Eng. 194(39-41), pp. 4135-4195, 2005. DOI: 10.1016/j.cma.2004.10.008

[3] R. Tielen, M. Möller, D. Göddeke, and C. Vuik: p-multigrid methods and their comparison to h-multigrid methods within Isogeometric Analysis. Comput. Methods Appl. Mech. Eng. 372, p. 113347, 2020. DOI: 10.1016/j.cma.2020.113347

Production projects:

Production projects should provide an additional 1-2 pages of documentation about how
(a) the code has been tuned to perform well on A64FX (ideally including benchmark data comparing performance with other architectures such as x86 or GPUs)
(b) it can make effective use of the key A64FX architectural features (notably SVE, the high-bandwidth memory, and NUMA characteristics)
(c) it can accomplish the scientific objectives within the available 32 Gbyte memory per node