LIBXSMM: A High Performance Library for Small Matrix Multiplications

Alexander Heinecke1, Hans Pabst2, Greg Henry3

1Intel Corporation, Intel Labs, Mission College Boulevard 2200, Santa Clara 95054, CA, USA
2Intel Semiconductor AG, Software and Service Group, Badenbusiness 39, 4000 Zürich, Switzerland
3Intel Corporation, Software and Service Group, 3111 NE 39th Avenue, Hillsboro 97124, OR, USA

Introduction

Small dense and small sparse matrix multiplications play an important role in modern high performance scientific applications, such as high-order FEM codes, block-sparse compressed row/column subroutines and sparse direct solvers using super-blocks, to mention just a few. Our proposed solution: LIBXSMM. The library generates code for the following instruction set extensions: Intel SSE3, Intel AVX, Intel AVX2. IMCI (KNCs) for Intel Xeon Phi coprocessors (“KNC”), and Intel AVX-512 as found in the future Intel Xeon Phi processor family and future Intel Xeon processors. Hereby a small problem is characterized by the $M$, $N$ and $K$ parameter of the corresponding matrix-matrix multiplication. LIBXSMM is best suited for problems sizes where $\sqrt{MNK} < 80$. LIBXSMM is available as free software at https://github.com/hfp/libxsmm.

General Design and Interface

LIBXSMM achieves its high application-level performance by a modular design providing a frontend (high level language interface, and routine selection), and a backend part (application specific xGEMM code generation).

Performance


Our test platform is a dual-socket Intel Xeon E5-2699v3 (“Haswell”) machine with 36 cores reaching 118 GB/s memory bandwidth in STREAM Triad, and 11 TFLOPS compute performance with large DGEKMIs. All measurements are based on Version 1.0 of LIBXSMM, https://github.com/hfp/libxsmm/releases.

CP2K

The performance depends on the workload, and therefore our results consist of 386 important code specializations which are binning into three groups of different problem sizes.

1. Specialized routine (implemented in assembly code), 2. Inlinable C/C++ code or optimized FORTRAN code, or 3. BLAS library call (fallback).

All three levels are directly accessible allowing to customize the mechanism (Fig. 2). The library also allows to amortize the cost of the dispatch when multiple calls with the same $M$, $N$ and $K$ are needed. Moreover, the threshold determining when to fallback into the BLAS implementation can be adjusted.

SeisSol

The earthquake code SeisSol is based on a high-order discontinuous Galerkin (DG) method. Its cell-local routines boil down to small sparse and dense matrix multiplications. The performance plots show one of the most important routines (multiplication with stiffness and flux matrices) and full application performance for several convergence orders.

Implementation

Frontend

LIBXSMM implements a three-level dispatch mechanism which helps executing the potentially best-performing implementation by finding specialized code for a particular problem instance $(M,N,K)$:

1. Specialized routine (implemented in assembly code),
2. Inlinable C/C++ code or optimized FORTRAN code, or
3. BLAS library call (fallback).

Assembly Backend

The assembly generator’s implementation follows ideas well-known in large xGEMMs, but due to the small sizes we cannot employ copy routines to build optimal $A$ or $B$ panels. Therefore it derives several highly performing micro-kernels which are orchestrated to form a small xGEMM operation. Specifically, our backend relies and on set of $(A,B,C)$:

- $S$ stands for scalar execution and $V$ for vector-register length. This ranges from 2 to 8 depending on instruction set and precision.

Summary/Outlook

LIBXSMM ...

- achieves optimal performance for wide range of small matrix multiplications on latest Intel Xeon E5v3 processors.
- achieves much better performance with respect to DRAM bandwidth and peak FLOPs than vendor libraries.
- is freely available and includes examples for usage.

Current Research:
- adding a runtime auto-tuning component for both dispatching and micro-kernel composition

References