Code Modernization: Application Specific Code Tuning on Intel Architecture

A case study from "Reasonable" up to "Closing the Ninja Gap"

Intel High Performance and Throughput Computing (EMEA)
Hans Pabst, July 9th 2015

Motivation

"Improving Performance by Specializing Code."

- Generating* specialized code where "everything is hard/hand-coded"
- Emit specialized code (optimized, Intrinsics, or assembly code)

* For example, via scripting such as Python (printed code is covering important cases), or at runtime (JIT).

Intel MKL

DIRECT CALL

Intel Math Kernel Library (Intel MKL)

DIRECT CALL feature: Intel MKL allows inlining code for very small problem sizes as well as calling the low-level library implementation directly for small problem sizes. This feature may improve performance because of skipping error checks and calls to intermediate library layers.

- Works for Intel and non-Intel compilers
- Works for C/C++ and Fortran
- Compile-time decision

Improved performance for small problem sizes.

Intel MKL 11.2: DIRECT CALL

Tradeoffs and Limitations

BLAS-conformant error checking vs. low overhead

- No error checking or 'xerbla' callback

Code dispatch vs. compile-time decision

- AVX, AVX2, no MIC code path

Subset of functions LAPACK/BLAS3

- xGEMM

Make informed tradeoffs and gain performance.

Intel MKL

VERBOSE MODE

Intel MKL 11.2: VERBOSE Mode

Quickly check if an application performs small matrix multiplications, and estimate the performance impact.

1. Set the environment variable MKL_VERBOSE=1
2. Select a representative workload
3. Run with redirected standard output

Example: evaluate occurrences of DGEMM for M, N, and K

```
$ env MKL_VERBOSE=1 ./myapplication > verbose.txt
$ grep -a "MKL_VERBOSE DGEMM" verbose.txt | cut -d, -f3-5
```

LIBXSMM

LIBXSMM

Interface (C API)

Simplified interface for matrix-matrix multiplications

- $c_{m \times n} = c_{m \times n} + a_{m \times k} * b_{k \times n}$ (no full xGEMM)

Dispatched and non-dispatched code paths

- Specialized/generated, or inlined C code
- LAPACK/BLAS (fallback code path)
- Amortized dispatch (“zero” overhead)

License

- Open Source Software (BSD 3-clause license)*

* https://github.com/hfp/libxsmm

Compiler Hints

Compiler Hints: Pragmas and Directives

Unknown pragmas (C/C++) or unknown directives (FORTRAN) are ignored if unknown to the specific compiler. However…

Preprocessor can unify similar directives across different compilers

BIG ISSUE: C/C++ preprocessor does not allow pragmas as part of macros

- C++: vendor specific compiler intrinsic (__pragma, etc.)
- C99: _Pragma intrinsic

What's worth to be covered / beneficial?

- PRAGMA_SIMD, PRAGMA_SIMD_COLLAPSE, PRAGMA_SIMD_REDUCTION
- PRAGMA_LOOP_COUNT, PRAGMA_UNROLL_N, PRAGMA_UNROLL
- ALIGNED, ASSUME_ALIGNED, ASSUME

Downloading a starting point…

https://github.com/hfp/libxsmm/blob/master/include/libxsmm_macros.h

Legal Disclaimer & Optimization Notice

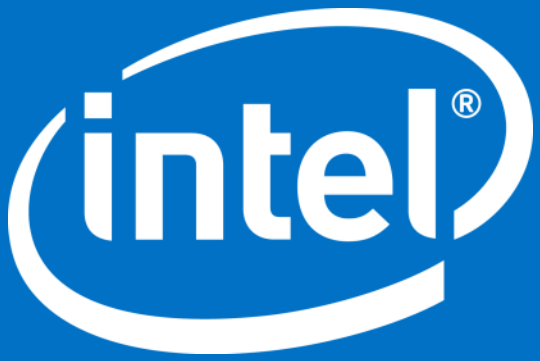
INFORMATION IN THIS DOCUMENT IS PROVIDED “AS IS”. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO THIS INFORMATION INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Optimization Notice

Intel’s compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Notice revision #20110804


intel®