Standardization of the Batched BLAS

Sven Hammarling

The University of Manchester

sven.hammarling@btinternet.com

Contributions from many colleagues:

icl.utk.edu/bblas/

SIAM CSE19, 26th February 2019

www.nlafet.eu
A Brief BLAS History

1970s  Level 1 BLAS developed, aimed at vector operations. Made the AXPY operation famous!
1979  Level 1 BLAS published in ToMS. The LINPACK Users’ Guide published
1988  Level 2 BLAS published in ToMS. Aimed at matrix-vector operations
1990  The Level 3 BLAS published in ToMS. Aimed at matrix-matrix operations
1992  LAPACK released on Gene Golub’s 15th birthday (February 29th)

The BLAS and LAPACK are still extensively used.
What Are the Batched BLAS (BBLAS)?

A BBLAS is intended to independently perform a large number of a specific BLAS operation. For example

\[ C_i \leftarrow \alpha_i A_i B_i + \beta_i C_i, \quad i = 1, 2, \ldots \ell, \]

where \( A_i, B_i \) and \( C_i \) are small matrices.

Many of the BLAS will have a corresponding BBLAS.
Functionality (e.g. $C_i \leftarrow \alpha_i \text{op}(A_i)\text{op}(B_i) + \beta_i C_i$, where $\text{op}(X)$ is $X$ or $X^T$ or $X^H$)

Description of routine interface (API)

Numerical stability

As with the BLAS.
Community Involvement

- For the current proposals there have been two workshops, in 2016 and 2017. The workshop reports can be found under: eprints.ma.man.ac.uk by searching for 2494 and 2543
- There were BoF sessions at SC17, ISC18 and SC18
- There was a two part minisymposium on the BBLAS and a presentation as part of an NLAFET minisymposium at SIAM PP18
- This two part BBLAS minisymposium here at SIAM CSE19.
- A report, D7.6 (Batched BLAS Specification), can be found at: www.nlafet.eu and under ‘PAPERS’ at: icl.utk.edu/bblas
Who Supports the BBLAS?

- Intel (MKL)
- Nvidia (cuBLAS)
- MATLAB (pagefun)
- MAGMA
- Kokkos
- Eigen
- ARM (when there is a standard)
- ...

Sven Hammarling
The matrices are gathered into same sized groups

Unlike the BLAS, an info argument is included and is input/output

A batch of matrices, for example the $A_i$, is passed in a single array, for example $A = (A_1 A_2 \ldots A_k)$

The BLAS name is appended by _batch

The principal language is C (it was Fortran for the BLAS)

A number of other proposals, such as a compact (interleaved) format for the arrays, were considered, but consensus led to the current standard.
The groups are determined by the integer argument $\text{group\_count}$ and the array of integers $\text{group\_sizes}$ for which the total number of matrices, $\text{batch\_count}$, is given as

$$\text{batch\_count} = \sum_{i=0}^{\text{group\_count}-1} \text{group\_sizes}(i).$$

(In C style.)
The interface for batched \texttt{dgemm}, for example, looks like:

\begin{verbatim}
cblas_dgemm_batch( group_count, *group_sizes, layout,
    *transA, *transB, *m, *n, *k,
    *alpha, **arrayA, *ldA,
    **arrayB, *ldb,
    *beta, **arrayC, *ldC,
    *info )
\end{verbatim}
Error handling is determined by the integer array info. On entry info(0) should have one of the following values:

**BblasErrorsReportAll**  All errors will be reported on output. Length of the array should be at least `batch_count`

**BblasErrorsReportGroup**  A single error from each group will be reported. Length of the array should be at least `group_count+1`

**BblasErrorsReportAny**  Occurrence of an error will be reported by a single integer value. Length of the array should be at least 1

**BblasErrorsReportNone**  No error will be reported on output. Length of the array should be at least 1

In the BLAS errors were managed by the routine XERBLAS.
Half and Quad Precision

We are adding half and quad precision to the specification. Half precision is important for many of the applications.

<table>
<thead>
<tr>
<th></th>
<th>Significand</th>
<th>Exponent</th>
<th>Approx. $x_{\text{max}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Half</td>
<td>11 bits</td>
<td>5 bits</td>
<td>$6.55 \times 10^4$</td>
</tr>
<tr>
<td>Single</td>
<td>24 bits</td>
<td>8 bits</td>
<td>$3.4 \times 10^{38}$</td>
</tr>
<tr>
<td>Double</td>
<td>53 bits</td>
<td>11 bits</td>
<td>$1.8 \times 10^{308}$</td>
</tr>
<tr>
<td>Quad</td>
<td>113 bits</td>
<td>15 bits</td>
<td>$1.2 \times 10^{4932}$</td>
</tr>
</tbody>
</table>

Table: IEEE Arithmetic Formats

We hope that BLAS implementers will also follow suit.
Because of the restricted range, half precision requires care. See for example Higham, Pranesh and Zounon, 2018. See also the first talk at BBLAS part II, proposing a next generation BLAS. (Includes half, quad and mixed precision.)

We don't propose that the BBLAS scale, but rather that it is done at a higher level.
Publication Process

- *De Facto* standard
- Aim to publish in ACM ToMS (as with the BLAS)
- Test software and reference implementation (also described in ToMS)
- The reference (or model) implementation is not intended to be efficient

The current reference software is available at: `github.com/NLAFET/BBLAS`
Reference implementation of the Batched BLAS routines (Group API).

116 commits 1 branch 0 releases 2 contributors

SrikaraPranesh Small changes to README
compute Indentations in compute
control add info parameter to command line arguments
core minot modification of comments in core
docs/doxygen add docs
include working zyryk tester
test Small changes in README
tools rm tools/subs.pyc
Makefile clean bblas_types.h, add control/constants.c, rm include/auxiliary.h
README.md Small changes to README
make.inc correction of bugs in zhemm

Batched Basic Linear Algebra Subroutines
University of Manchester (UK)
University of Tennessee (US)

Download BBLAS Software
For the future, we can expect the development of batched LAPACK routines.

Indeed, the MAGMA package, aimed at heterogeneous architectures, is an example that has already implemented some batched LAPACK routines, particularly for the solution of $Ax = b$.

See: icl.cs.utk.edu/magma and the next talk in this session!