SC24 Batch Linear Algebra Software, Present and Future (Birds of a Feather)

Evolution of Batched Linear Algebra in Intel® oneAPI Math Kernel Library (Intel oneMKL)

Sarah Knepper

intel

Group Batched APIs

- First introduced: Intel MKL 11.3 (C/Fortran); oneMKL 2021.1 (SYCL)
- Operates on multiple groups (a number of operations with same parameters) simultaneously
- Two additional parameters compared to non-batched function: group_count and group_size (array)
- Consistent level of redirection for parameters: integer becomes array of integers; matrix pointer becomes array of matrix pointers
- Example C API:

```
void cblas_dgemm_batch (const CBLAS_LAYOUT Layout,
const CBLAS_TRANSPOSE* transa_array, const CBLAS_TRANSPOSE* transb_array,
const MKL_INT* m_array, const MKL_INT* n_array, const MKL_INT* k_array,
const double* alpha_array,
const double **a_array, const MKL_INT* lda_array,
const double **b_array, const MKL_INT* ldb_array,
const double* beta_array,
double **c_array, const MKL_INT* ldc_array,
const MKL_INT group_count, const MKL_INT* group_size);
```

Strided Batched APIs

- First introduced: oneMKL 2021.1 (C/Fortran/SYCL)
- All matrices in the batch share the same parameters
- Additional parameters compared to non-batched function: batch_size and strides
- All matrices are stored at constant stride from each other
- Example C API:

```
void cblas_dgemm_batch_strided (const CBLAS_LAYOUT layout,
const CBLAS_TRANSPOSE transa, const CBLAS_TRANSPOSE transb,
const MKL_INT m, const MKL_INT n, const MKL_INT k, const double alpha,
const double *a, const MKL_INT lda, const MKL_INT stridea,
const double *b, const MKL_INT ldb, const MKL_INT strideb,
const double beta,
double *c, const MKL_INT ldc, const MKL_INT stridec,
const MKL_INT batch_size);
```

Span Inputs

- First introduced: oneMKL 2021.4 (SYCL)
- For gemm_batch group SYCL APIs, we also support span inputs
- Span can be of size 1, the number of groups, or the total batch size:
  - Size 1: parameter reused for all gemm computations
  - Size group count: parameter reused for all gemm within a group
  - Size total batch count: each gemm computation uses a different value for this parameter
- Example SYCL API:

```
sycl::event gemm_batch(sycl::queue &queue, const
sycl::span<oneapi::mkl::transpose> &transa, const
sycl::span<oneapi::mkl::transpose> &transb, const sycl::span<std::int64_t> &m,
const sycl::span<std::int64_t> &n, const sycl::span<std::int64_t> &k,
const sycl::span<double> &alpha, const sycl::span<const double*> &a,
const sycl::span<std::int64_t> &lda, const sycl::span<const double*> &b,
const sycl::span<std::int64_t> &ldb, const sycl::span<double> &beta,
sycl::span<double*> &c, const sycl::span<std::int64_t> &ldc, size_t group_count,
const sycl::span<size_t> &group_sizes, compute_mode mode = compute_mode::unset,
const std::vector<sycl::event> &dependencies = {})
```

Compact APIs

- First introduced: Intel MKL 2018 (C)
- Functionality: gemm, trsm, getrfnp, getrinp, potrf, geqrf
- All matrices in the batch share the same parameters
- Additional parameters compared to non-batched: format and batch size (nm)
- Requires matrices to be stored in SIMD-friendly format (interleaved elements)
- Best for tiny sizes
- Supported only on CPU
- Example C API:

```
void mkl_dgemm_compact (MKL_LAYOUT layout, MKL_TRANSPOSE transa, MKL_TRANSPO
SE transb, MKL_INT m, MKL_INT n, MKL_INT k, double alpha, const double *ap,
MKL_INT ldap, const double *bp, MKL_INT ldbp, double beta, double *cp, MKL_I
NT ldcp, MKL_COMPACT_PACK format, MKL_INT nm);
```

Batched Functionality Available in Intel oneMKL

- BLAS Level-1 Related: axpy, copy, imatcopy, omatcopy, omatadd
- BLAS Level-2 Related: gemv, dgmm
- BLAS Level-3 Related: gemm, gemm3m, syrk, trsm
- LAPACK LU Related: getrf, getrfnp, getrs, getrsnp, getri{_oop}, geinv
- LAPACK QR Related: geqrf, orgqr, ungqr, gels
- LAPACK Cholesky Related: potrf, potrs
- LAPACK SVD Related: gesvda

Note that not all APIs are supported for all functionality

Questions

- Are group or strided APIs more useful?
- Would span inputs be useful for more batched functionality?
- What batched functionality is most useful?
  - Anything you'd like to see (e.g., gesv)?
- What programming API do you prefer (C, Fortran, SYCL)?


intel