oneMKL Technical Advisory Board

Session 5

July 15, 2020

Agenda

- Welcoming remarks – 5 minutes
- Discussion and updates from last meeting – 15 minutes
- Overview of oneMKL Random Number Generators domain – Pavel Dyakov and Alina Elizarova (30 minutes)
- Wrap-up and next steps – 5 minutes

oneMKL TAB Members

- Mehdi Goli, Codeplay
- Edward Smyth, Numerical Algorithms Group (NAG)
- *Mike Dewar, NAG* - stepped down

- Mark Hoemmen, Stellar Science
- Nevin Liber, Argonne National Laboratory (ANL)
- Piotr Luszczek, Innovative Computing Laboratory (ICL) at University of Tennessee, Knoxville (UTK)
- Pat Quillen, MathWorks
- Nichols Romero, ANL
- Harry Waugh, University of Bristol

- Brief intro: your job; how you use math libraries

- Have you reviewed the oneMKL spec?
- Any additional comments?

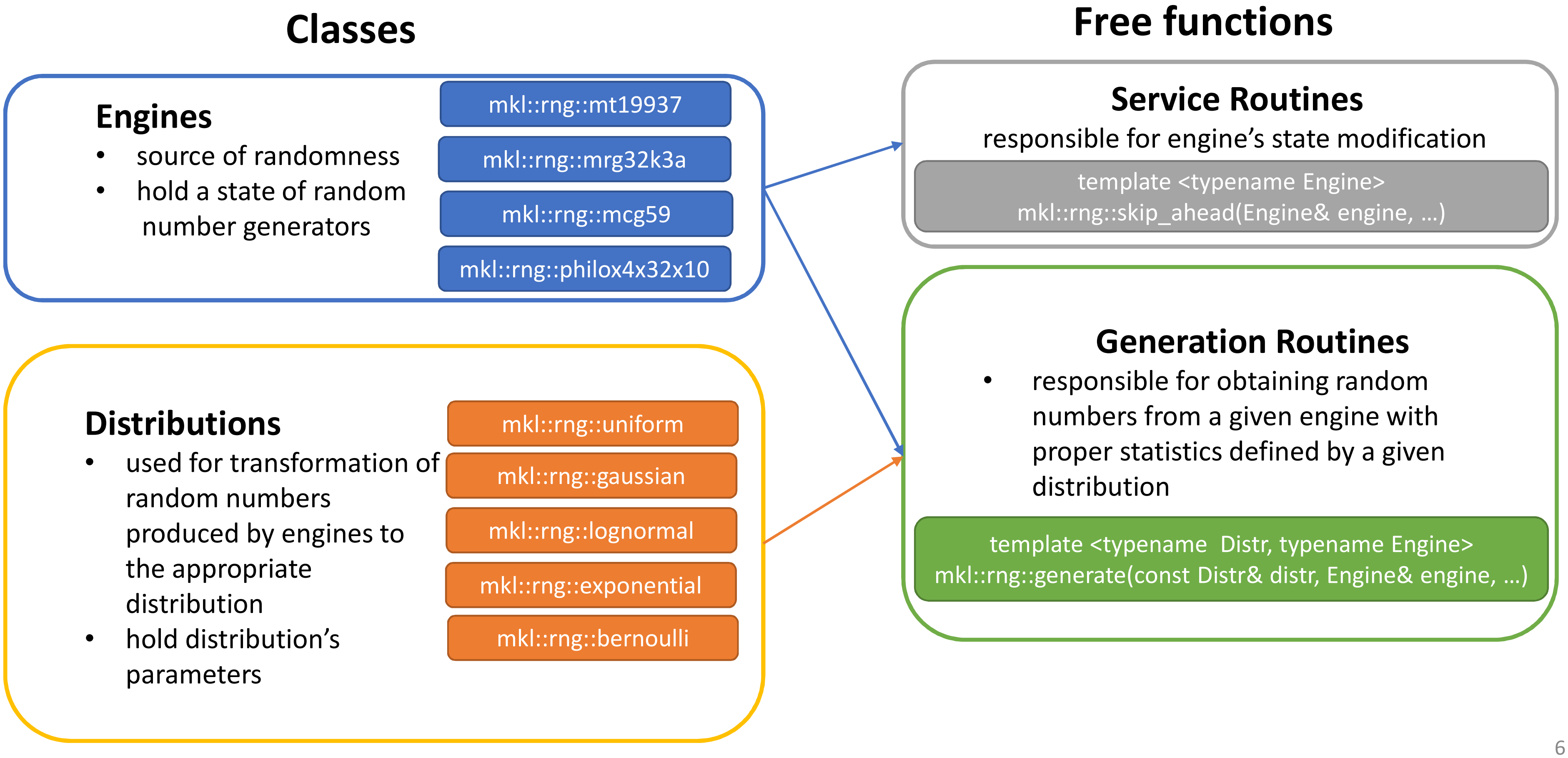
Row-Major Support for BLAS

- For convenience in slides: `namespace mkl = oneapi::mkl;`

- Adding row-major support for BLAS domain via namespaces:
  - Add `mkl::blas::row_major` namespace: contains all BLAS entry point and assume matrices are stored using row major layout
    `mkl::blas::row_major::gemm(ta,tb,m,n,k,alpha,a,lda,b,ldb,beta,c,ldc)`
  - Add `mkl::blas::column_major` namespace: contains all BLAS entry point and assume matrices are stored using column major layout
    `mkl::blas::column_major::gemm(ta,tb,m,n,k,alpha,a,lda,b,ldb,beta,c,ldc)`
  - Keep all BLAS entry point in `mkl::blas` namespace with default column major layout
    `mkl::blas::gemm(ta,tb,m,n,k,alpha,a,lda,b,ldb,beta,c,ldc)`
- Will revisit this for version 1.0+

Overview of oneMKL Random Number Generators (RNG) Domain

RNG Structure



Classes

Engines

- source of randomness
- hold a state of random number generators

mkl::rng::mt19937

mkl::rng::mrg32k3a

mkl::rng::mcg59

mkl::rng::philox4x32x10

Distributions

- used for transformation of random numbers produced by engines to the appropriate distribution
- hold distribution's parameters

mkl::rng::uniform

mkl::rng::gaussian

mkl::rng::lognormal

mkl::rng::exponential

mkl::rng::bernoulli

Free functions

Service Routines

responsible for engine's state modification

```
template <typename Engine>
mkl::rng::skip_ahead(Engine& engine, …)
```

Generation Routines

- responsible for obtaining random numbers from a given engine with proper statistics defined by a given distribution

```
template <typename  Distr, typename Engine>
mkl::rng::generate(const Distr& distr, Engine& engine, …)
```

RNG Usage Model

- Create and initialize an engine object.
    - Engine’s state can be adjusted by service functions if required.
- Create and initialize a distribution object.
- Call the generate routine to obtain random numbers with appropriate statistics properties.

```cpp
#include <vector>
#include "CL/sycl.hpp"
#include "mkl_rng_sycl.hpp"

int main() {
   sycl::queue queue;
   const size_t n = 10000;
   const uint64_t seed = 1234;
   std::vector<double> r(n);
   // create engine object
   mkl::rng::philox4x32x10 engine(queue, seed);
   // create distribution object
   mkl::rng::gaussian<double> distr(0.0, 1.0);
   {
      sycl::buffer<double, 1> r_buf(r.data(), r.size());
      // fill r_buf with random numbers
      mkl::rng::generate(distr, engine, n, r_buf);
   }
   return 0;
} // *
```

*Both mkl::rng:: and oneapi::mkl::rng:: are considered to be used for oneMKL RNG functionality

RNG Engines Classes

- Represent source of independent and identically distributed random variables.
- Independent. Represent different algorithms:
  - mt19937 – Mersenne Twister generator.
  - mcg31m1 – Multiple-congruential generator.
- Hold sycl::queue object – generation is performed on the device associated with the queue.

```
namespace mkl {
namespace rng {

class philox4x32x10 {
public:
    philox4x32x10(sycl::queue& queue, std::uint64_t seed); // *

    philox4x32x10(sycl::queue& queue,
                  std::initializer_list<std::uint64_t> seed); // *

    philox4x32x10(const philox4x32x10& other);

    philox4x32x10& operator=(const philox4x32x10& other);

    ~philox4x32x10();
};

} // namespace rng
} // namespace mkl
```

* Engine-defined constructors

RNG Distributions Classes Templates

- Represent the statistical properties of the produced random numbers.
- Independent. Each distribution may be combined with each engine*.
- Template parameters:
  - Type of random numbers.
  - Method of transformation (e.g. icdf).
  - Other: optional.
- Run-time parameters - specific for each distribution (e.g. mean and standard deviation for Gaussian).

```
namespace mkl {
namespace rng {

namespace gaussian_method {
    struct icdf{};
    struct box_muller{};
    struct box_muller2{};
    using by_default = icdf;
}

template<typename RealType = float,
         typename Method = gaussian_method::by_default>
class gaussian {
public:
    using method_type = Method;
    using result_type = RealType;

    gaussian();

    explicit gaussian(RealType mean, RealType stddev);

    explicit gaussian(const gaussian<RealType, Method>&);

    RealType mean() const;

    RealType stddev() const;

    gaussian<RealType, Method>& operator= (
                                const gaussian<RealType, Method>&);
};
} // namespace rng
} // namespace mkl
```

*exception: not all engines support uniform_bits distribution – to provide uniformly distributed 32- or 64-bits chunks

RNG Generation Routines

- Routines to provide random numbers with statistics of a given distribution using a given engine as a source of randomness.
- Use parameter `sycl::buffer<>` or USM pointer `r`, provided by user as a storage for random numbers.

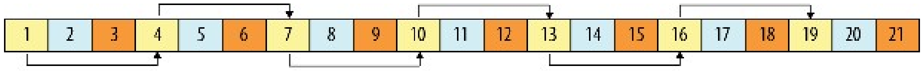
```
namespace mkl {
namespace rng {

template<typename Distr, typename Engine>
void generate(const Distr& distr, Engine& engine, std::int64_t n,
    sycl::buffer<typename Distr::result_type, 1>& r);

template<typename Distr, typename Engine>
sycl::event generate(const Distr& distr, Engine& engine, std::int64_t n,
    typename Distr::result_type* r, const sycl::vector_class<sycl::event> &dependencies = {});

} // namespace rng
} // namespace mkl
```

RNG Service Routines

- Service routines are used to modify state of engine.
- Represented as free functions.

```
namespace mkl {
namespace rng {

template<typename Engine>
void skip_ahead(Engine& engine, std::uint64_t num_to_skip);

template<typename Engine>
void skip_ahead(Engine& engine,
     std::initializer_list<std::uint64_t> num_to_skip);

template<typename Engine>
void leapfrog(Engine& engine, std::uint64_t idx, std::uint64_t stride);

} // namespace rng
} // namespace mkl
```

Leapfrog model



At the 1st node, the engine generates: 1, 4, 7, 10, 13, 16, 19.

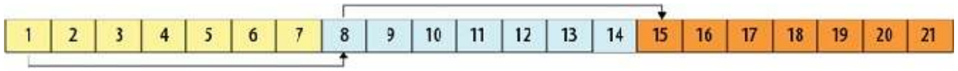
At the 2nd node, the engine generates: 2, 5, 8, 11, 14, 17, 20.

At the 3rd node, the engine generates: 3, 6, 9, 12, 15, 18, 21.

Legend:

🟨 1st node engine
🟦 2nd node engine
🟧 3rd node engine

Skip-ahead model



At the 1st node, the engine generates: 1, 4, 7, 10, 13, 16, 19.

At the 2nd node, the engine generates: 2, 5, 8, 11, 14, 17, 20.

At the 3rd node, the engine generates: 3, 6, 9, 12, 15, 18, 21.

Legend:

🟨 1st node engine
🟦 2nd node engine
🟧 3rd node engine

Next Steps

- Look over current oneMKL Spec v. 0.8.5
  - Changes from v0.8: Refactor gemm/gemm_ext with gemm_bias; formatting
  - Updated APIs for RNG and other domains to appear in v0.9
- Focuses for next meeting(s):
  - Additional APIs

Version	Date	oneAPI Notes	oneMKL Notes
0.8.5	26 June 2020	80% content	Finalize BLAS and LAPACK domains
0.9.0	30 Jul 2020	~100% content	Finalize FFT, sparse BLAS, RNG, and VM domains
1.0.0	30 Aug 2020	Gold Release	Minor cleanup

Resources

- oneAPI Main Page: https://www.oneapi.com/
- Latest release of oneMKL Spec (currently v. 0.8.5): https://spec.oneapi.com/versions/latest/elements/oneMKL/source/index.html
- GitHub for oneAPI Spec: https://github.com/oneapi-src/oneAPI-spec
- GitHub for oneAPI TAB: https://github.com/oneapi-src/oneAPI-tab
- Latest build of oneAPI Spec: http://staging.spec.oneapi.com.s3-website-us-west-2.amazonaws.com/exclude/ci/branches/refs/heads/master/versions/latest/index.html

- GitHub for open source oneMKL interfaces (currently BLAS domain): https://github.com/oneapi-src/oneMKL