Portable Performance in Numerical Linear Algebra Software with OpenMP

Piotr Luszczek
University of Tennessee

Innovative Computing Laboratory
PLASMA: Parallel Linear Algebra for Scalable Multicores & Accelerators

- Dense: linear, least-squares, EIG/SVD (with vectors)
- Tile matrix layout and OpenMP 4 tasking
- Threading variants
  - POSIX (obsolete)
  - WinThreads (obsolete)
  - OpenMP 4+ (currently supported)
- Targeted Compilers and Software Stacks
  - OSS
    - LLVM Clang 11, GNU GCC 10+
  - Accelerator vendors
    - AMD AOMP 11, Intel OneAPI, NVIDIA NVHPC SDK 20+
  - Integrators
    - IBM XL 16, HPE/Cray 9
PLASMA with OpenMP Tasking and Offload

- Multicore
  - task
  - taskloop

- Accelerators
  - Data transfers
  - Offload kernels
PLASMA Task Scheduling

- task 1
  - core 1
- task 2
  - task 3
  - core 2
- task 4
  - core 3

- GPU 1
  - GPU 2
On-Device Performance From Vendor Libraries

```c
int dev = omp_get_default_device();
double *a = mkl_malloc(a_ld * n * 8, 64);
#pragma omp target data map(to:a,b) map(tofrom:c)
{
    #pragma omp target variant dispatch use_device_ptr(a,b,c) device(dev) \
    nowait
    mkl_dgemm(tA, tB, m, n, k, alpha, a, a_ld, b, b_ld, beta, c, c_ld);
    #pragma omp taskwait
}
```

- Common interfaces that Reuse or emulate established interfaces (optimized by vendors)
  - DGEMM(), cuDgemm(), hipDgemm(), rocDgemm(), mkl_dgemm()
On-Device Storage with Device-Resident Allocations

```c
double *a = omp_target_alloc(device, a_ld * n * 8, 64);
omp_target_memcpy(..., a);
omp_target_free(a);
```

- Abstractions that are well defined for practical structure of objects formed from users’ data
  - Focusing on user experience
  - Object hierarchy for matrix, vector, execution policy (host or device)
#pragma omp target data map(a[0:n*n],b[0:n*n]) map(alloc:c[0:n*n])
#pragma omp target data use_device_ptr(a,b,c)
{
    cudaStream_t ompStream = (cudaStream_t)omp_get_cuda_stream(dev);
    cublasSetStream(handle, stream);
    cublasDgemm(handle, CUBLAS_OP_N, CUBLAS_OP_N, m, n, k,
             &alpha, a, a_ldim, b, b_ld, &beta, c, c_ldim);
}
SLATE: Scalable Linear Algebra Targeting Exascale

- Compute targets
  - Distributed memory
  - Multicore
  - Multi-GPU systems
- Flexible tile storage with affinity tracking
- Generic algorithms
  - Programming against generic types
  - Testing on concrete types
SLATE: dense and low-rank algorithms

- Large scope in terms of hardware
  - Multicore
    - ARM, POWER, x86
  - GPUs
    - CUDA, HIP, SYCL
  - Distributed memory
    - MPI (multithreaded)

- Large algorithmic scope
  - Dense algorithms: linear, least-squares, eigenvalue, SVD
  - Matrix types and storages: rectangular, square, triangular, trapezoidal
  - Low-rank algorithms (ACA, low-rank tiles, …)

- OpenMP coordinates with MPI, use of cores, and launching kernels on devices
Reusable Portability Layers

- Dense and batched kernels hiding HPC compute code
  - BLAS++
  - LAPACK++
- Randomized kernels and algorithms hiding HPC randomization, projections, and iteration
  - RandBLAS
  - RandLAPACK