Log

Contents

1. Env 2
2. examples/00_basic_gemm 2
    2.1. Build 2
3. examples/01_cutlass_utilities 2
4. examples/02_dump_reg_shmem 2
5. examples/05_batched_gemm 2
6. examples/06_splitK_gemm □ 3
7. examples/cute/tutorial/sgemm_1.cu 3
8. media/docs/cute/01_layout.md 3

1. Env

- Commit: `5c447dd84f8ae0e1d48ff9a2eae26ce8c4958101`
- Docker: `docker.io/nvidia/cuda:12.0.1-devel-ubuntu22.04`
  - install `cmake`, `python` and `git` manually
  - `export CUDACXX=$(which nvcc)`

2. examples/00_basic_gemm

2.1. Build

The example is at `examples/00_basic_gemm`.

```
# entry point is the root of the repository
mkdir build && cd build
# -DCUTLASS_LIBRARY_KERNELS=basic_gemm where `basic_gemm` is the name of the file (basic_gemm.cu)
cmake .. -DCUTLASS_NVCC_ARCHS=80 -DCUTLASS_LIBRARY_KERNELS=basic_gemm
cd examples/00_basic_gemm
# build the 00_basic_gemm excutable
make
```

2024-04-26

3. examples/01_cutlass_utilities

This example uses cutlass utils to create a random GPU GEMM operation and compare the results with the CPU reference implementation.

Several utils in this example

- `cutlass::half_t`
- `cutlass::HostTensor<>`
- `cutlass::reference::device::TensorFillRandomGaussian()`
- `cutlass::reference::host::Gemm<>`
- `cutlass::reference::host::TensorEquals()`

2024-06-10

4. examples/02_dump_reg_shmem

- Dump a matrix

```
int main() {
  // Initialize a 64x32 column major matrix with sequential data (1,2,3...).
  using Element = cutlass::half_t;
  using Layout = cutlass::layout::ColumnMajor;

  cutlass::HostTensor<Element, Layout> matrix(
      {EXAMPLE_MATRIX_ROW, EXAMPLE_MATRIX_COL});
  cutlass::reference::host::BlockFillSequential(matrix.host_data(),
                                                matrix.capacity());

  // Dump the matrix.
  std::cout << "Matrix:\n" << matrix.host_view() << "\n";
  // ...
}
```

  - Template `cutlass::reference::host::BlockFillSequential` is similar to `np.arange(NUM_ELEMENTS)` in Python.

- TODO: dump SMEM

5. examples/05_batched_gemm

Batched GEMM between two column major matrices.

```
This example demonstrates how to use cutlass to compute a batched strided gemm in two different ways:
  1. By specifying pointers to the first matrices of the batch and the stride between the consecutive
     matrices of the batch (this is called a strided batched gemm).
  2. By copying pointers to all matrices of the batch to the device memory (this is called an array
gemm).
In this example, both A and B matrix are non-transpose and column major matrix
batched_C = batched_A x batched_B

batched_A = [A_batch1 A_batch2]
batched_B = [[B_batch1], [B_batch2]]
batched_C = [C_batch1 C_batch2]
```

2024-06-10

6. examples/06_splitK_gemm ⭐

This example is **well-documented**.

- Naming Convention of GEMM template types: `D = alpha * A * B + beta * C`
  - `ElementComputeEpilogue` : epilogue operations
  - `ElementInputA` : `A`
  - `ElementInputB` : `B`
  - `ElementAccumulator`
  - `ElementOutput` : `D`
- `MxNxK` : numbers like 128x128x32 refer to the tile size (`MxNxK`) of threadblock, warp, or mma-op.
- Get and check device property: `cudaGetDeviceProperties`
- `cutlass::gemm::device::GemmSplitKParallel`

7. examples/cute/tutorial/sgemm_1.cu

- Prerequisite
  - go through `../media/docs/cute/01_layout.md` (Section 8)

2024-04-26

8. media/docs/cute/01_layout.md

Keywords: `layout` , `shape` , `stride` , `coordinate`

- layout = (shape, stride)
- coordinate:
  - `cute::idx2crd(idx, shape)` maps index/input coordinate to natural coordinate via `shape`
  - `cute::crd2idx(crd, shape, stride)` maps input coordinate/natural coordinate (an IntTuple) to index (an integer) via `shape` and `stride`