

Cuda Pipeline

pipeline

proxy pattern, `cuda::pipeline` thread `pipeline_shared_state` proxy

- `pipeline_shared_state` pipeline , barrier .
- `pipeline_shared_state`
- `thread_scope` pipeline , ,
`cuda::pipeline<cuda::thread_scope_thread> pipeline = cuda::make_pipeline()`
- **`make_pipeline`** , **`pipeline_shared_state`** **role**,
group **producer/consumer**
- pipeline **fifo, head in, tail out**, **pipeline** **stage**
- pipeline proxy , consumer, producer, both
- `fifo` `pipeline_shared_state` , `fifo` stage , producer
 acquire .
- `role` `producer` pipeline:
 - `pipeline.producer_acquire()`; thread `fifo` push , lock
 - `producer_acquire` , thread `async` acquire stage
 - `pipeline.producer_commit()`; thread `async`
 - `group` `producer` `commit` , stage `push` `fifo` , stage
 `async` ready
- `role` `consumer` pipeline
 - `pipeline.consumer_wait()`; thread `fifo` ,
 - `consumer_wait()` , `group` `producer` stage ready,
 - `pipeline.consumer_release()`; thread .
 - `group` `consumer` `release` , stage `pop` `fifo`
- : stage , `fifo` tail track

```
// pipeline API producer consumer
cuda::pipeline pipeline = cuda::make_pipeline(block, &shared_state, thread_role);
if (thread_role == cuda::pipeline_role::producer) {
    // Only the producer threads schedule asynchronous memcpys:
    pipeline.producer_acquire();
    size_t shared_idx = fetch_batch % stages_count;
    size_t batch_idx = fetch_batch;
    size_t global_batch_idx = block_batch(batch_idx) + thread_idx;
    size_t shared_batch_idx = shared_offset[shared_idx] + thread_idx;
```

```
cuda::memcpy_async(shared + shared_batch_idx, global_in + global_batch_idx, sizeof(int), pipeline);
// [ ] [ ] [ ] [ ]
pipeline.producer_commit();
}
if (thread_role == cuda::pipeline_role::consumer) {
    // Only the consumer threads compute:
    // [ ] [ ] [ ] [ ]
    pipeline.consumer_wait();
    size_t shared_idx = compute_batch % stages_count;
    size_t global_batch_idx = block_batch(compute_batch) + thread_idx;
    size_t shared_batch_idx = shared_offset[shared_idx] + thread_idx;
    compute(global_out + global_batch_idx, *(shared + shared_batch_idx));
    pipeline.consumer_release();
}
```

Revision #1

Created 11 January 2025 09:46:27 by Colin

Updated 12 January 2025 06:33:50 by Colin