The Search for Concurrency!

Across Multiple Hardware Platforms

with OpenMP and SYCL on GPUs of tasks

Thomas Applencourt

February 15, 2023

Introduction

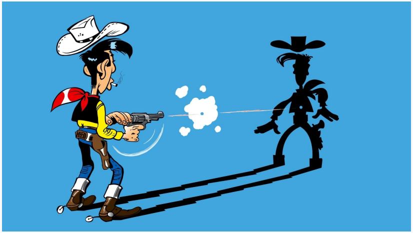
Disclaimer

- This is just a micro-benchmark! Do not over-extrapolate.
- This is a snapshot in time, with my current environment. Results will change!
- Didn't try that many compilers
- **Don't trust me, just measure[1] by yourself**

[1] *https://github.com/argonne-lcf/HPC-Patterns/tree/main/concurency*

What is OpenMP and SYCL?

We don't have time for That! If you don't know them, I will teach you during the talk.



More seriously...

1. SYCL and OpenMP argue to be portable programming model
2. We will verify that they are ”performance” portable [2] so that you can design your application accordingly

[2] For a tiny subset of behavior on A100, Mi250, PVC

What is Concurrency?

- ”Concurrency”: our tasks[3] can be executed out-of-order
- ”Parallelism” the fact that our task are really executing at the same time.

[3] Kernels, commands, programs, ...

General Goal of this Talk

Overlaps of computation and data-transfer is one of the 101 gpu-optimization. We will verify that OpenMP and SYCL in GPU can do it!

- We will explore how to express ”task concurrency” in SYCL and OpenMP[4]
- And verify is we achieve parallelism (we do HPC!)

We will **not** talk about concurrency inside kernel (work-item, threads,...). But parallelism between tasks!

[4] Using multiple MPI rank per GPU is lame...

More precise goal of this talk

We will see if we can overlap:

- Compute Kernels
- Compute kernel and Data-Transfers
- Bi-directional data-transfers

Motivation

Doing things in parallel is better than doing serially

- GPUs are large. You want to maximize the ”global occupancy” (how many compute units you are using)
    - You may have no choose than to run multiple kernels in parallel
- PCI is damn slow! You want to:
    - Saturate the Bandwidth! PCIs is a ”fully-duplex” protocol so do concurrent bidirectional transfers[5]
    - Overlaps compute and data-transfer

[5] if you are not doing it, you are wasting bandwidth

Argonne NATIONAL LABORATORY

Concurrency

OpenMP Concurrency[6]

Host Threads

```
#pragma omp parallel for
for (auto c: commands)
  #pragma omp target [...]
  {}

```

No wait

```
for (auto c: commands)
  #pragma omp target [...] nowait
  {}

#pragma omp taskwait
```

(The target region can be anything. A *target team distribute...* followed by a kernel, or a *target update*)

[6] Please implementer support meta-directive so I don't need to 'ifdef' my poor benchmark and have two binaries

Argonne NATIONAL LABORATORY

SYCL Concurrency

Out of order queue:

```
 sycl::queue Q;
for (auto& c: commands)
  do_work(Q, c);
Q.wait();
```

Pools of In order Queues:

```
const sycl::device D;
const sycl::context C(D);
std::vector<sycl::queue> Qs;
// Creating the in-order queues
for (auto _: commands)
  Qs.push_back(
    sycl::queue(
       C, D,
       sycl::property::queue::in_order{}
    );
  );
// Submitings jobs
for (int i = 0; i < commands.size(); i++)
  do_work(Qs[i], commands[i]);
for (auto &Q : Qs)
 Q.wait();
```

Measurement

Benchmark

- We will measure the time is take to perform N commands serially. Commands can be
    - Memcopy from Device memory to Host memory[7]
    - Memcopy from Malloced memory to Device
    - ...
    - Compute kernel
- Them we will measure the time is took to perform them concurrently
- Success is we have a N-x speed-up!

[7] Host memory = Pinned Memory

A little bit of technical details[10]

- We auto-tune the commands so they take the same times
  - Data-transfer payload a large (few hundreds of megabyte). We are reaching ”peak” BW.
  - The compute kernel use only one work-item / cuda-threads. But consist of large FMA chains[8]
- We run the experiment 200 times and take the min time[9]
- In openmp we first `enter-data` and then use `update` for the memcopy
- For OpenMP pinned memory: We used `omp_target_alloc_host` and try to use `llvm_omp_target_alloc_host`

[8] cl-peak like

[9] Principle of charity and this avoid dealing with all the JITing, power throttling, ... noises

[10] Just to make you think that I know what I'm talking about...

Example of logfile:

Small Log of a AMD run:

```
./sycl in_order --commands H2D D2H
Minimum Measured Total Time Serial: 69793us
  Minimum Time Command 0 ( HD): 34912us (28.6434 GBytes/s)
  Minimum Time Command 1 ( DH): 34881us (28.5405 GBytes/s)
Maximum Theoretical Speedup: 1.99911x
Minimum Measured Total Time //: 42465us (46.9922 GBytes/s)
Speedup Relative to Serial: 1.64354x
## in_order | HD DH | SUCCESS: Close from Theoretical Speedup
```

Success![11]

[11] I have kind of a ”easy” threshold for success, 30% of ideal

Nvdia A100: OpenMP

Using:

- *clang version 16.0.0 https://github.com/intel/llvm.git aa69e4d9b86*
- *cudatoolkit 11.8.0*

commands	host threads	nowait
C C	SUCCESS	FAILURE
C M2D	SUCCESS	FAILURE
C D2M	SUCCESS	FAILURE
H2D D2H	NOT RUN	NOT RUN
M2D D2M	FAILURE	FAILURE

- Was not able to get Host allocation to work
- Nowait -> OpenMP Runtime Issue

NVDIA A100: SYCL

Using:

- *clang version 16.0.0 https://github.com/intel/llvm.git 055ee225*
- *cudatoolkit 11.8.0*

commands	Qs in-order	Q out-of-order
C C	SUCCESS	SUCCESS
C M2D	SUCCESS	SUCCESS
C D2M	SUCCESS	SUCCESS
H2D D2H	kind of SUCCESS	Kind of SUCCESS
M2D D2M	FAILURE	FAILURE

Bidirectional bandwidth higher than unidirectional, but not in the 30% tolerance

AMD MI250: OpenMP

Using:

- *`AMD clang version 15.0.0 https://github.com/RadeonOpenCompute/llvm-project roc-5.4.0 22465 d6f0fe8`*
- *`rocm-5.4.0`*

commands	host threads	nowait
C C	SUCCESS	FAILURE
C M2D	SUCCESS	FAILURE
C D2M	SUCCESS	FAILURE
H2D D2H	NOT RUN	NOT RUN
M2D D2M	FAILURE	FAILURE

- Was not able to get Host allocation to work
- Nowait -> OpenMP Runtime Issue

AMD MI250: SYCL

Using:

- *clang version 16.0.0 (git@github.com:intel/llvm.git 0b1fd8df661)*
- *rocm-5.4.0*

commands	Qs in-order	Q out-of-order
C C	SUCCESS	SUCCESS
C M2D	SUCCESS	SUCCESS
C D2M	SUCCESS	SUCCESS
H2D D2H	SUCCESS	SUCCESS
M2D D2M	FAILURE	FAILURE

M2D, D2M Failure -> Not a SYCL issue. Lower level problem

Intel PVC: OpenMP

- Using OneAPI 2022.12.30.003
- *ZE_AFFINITY_MASK=0.0*
- *LIBOMPTARGET_LEVEL_ZERO_USE_IMMEDIATE_COMMAND_LIST=1*
- *LIBOMPTARGET_LEVEL0_USE_COPY_ENGINE=main*

commands	host threads	nowait
C C	SUCCESS	SUCCESS
C M2D	SUCCESS	SUCCESS
C D2M	SUCCESS	SUCCESS
H2D D2H	FAILURE	FAILURE
M2D D2M	FAILURE	FAILURE

H2D, H2M -> We are aware of the bug. Working with Intel to mitigate it.

Intel PVC: SYCL

- Using OneAPI 2022.12.30.003
- *ZE_AFFINITY_MASK=0.0*
- *SYCL_PI_LEVEL_ZERO_USE_IMMEDIATE_COMMANDLISTS=1*

commands	Qs in-order	Q out-of-order
C C	SUCCESS	SUCCESS
C M2D	SUCCESS	SUCCESS
C D2M	SUCCESS	SUCCESS
H2D D2H	FAILURE	FAILURE
M2D D2M	FAILURE	FAILURE

H2D, H2M -> We are aware of the bug. Working with Intel to mitigate it.

Conclusion

Conclusion

Good news:

- **Overlaps of compute/compute and compute/data-transfert work in all Hardware, all programming model!**
- intel/SYCL rocks in all the backend!
- If you want PCI bi-directional data-transfer concurrency one need need to use Host Memory[12]

Not so Good news:

- Currently PVC doesn't exploit PCI full-duplex capability
- OpenMP Nowait need some love on AMD, NVDIA
- I was not able to use host-memory on AMD, NVDIA

[12] Also know at pinned memory

Acknowledgment

Thanks for a lot of people who helped me with this talk (Colleen, Abhi, Aksel, ...).

This research was supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy (DOE) Office of Science and the National Nuclear Security Administration.

This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.