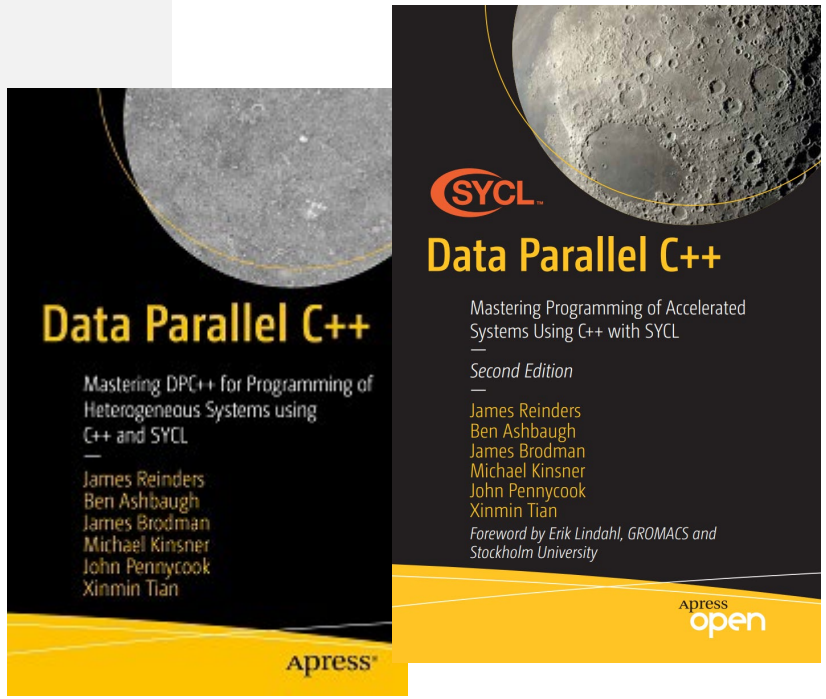
SYCL

https://github.com/jamesreinders/syclmultiple

James Reinders



intel®

Today

- What/why is C++ with SYCL ?
- Learn C++ with SYCL
- Assignment: write a program using C++ with SYCL that distributes work across multiple devices
  - Resources: our book, free tools from Intel, free cloud resources from Intel
- Q&A

Today

- What/why is C++ with SYCL ?
- Learn C++ with SYCL
- Assignment: write a program using C++ with SYCL that distributes work across multiple devices
  - Resources: our book, free tools from Intel, free cloud resources from Intel
- Q&A

A word about C++

It’s not a Python vs. C++ vs. C vs. Fortran world. They are tools.

What is C++ with SYCL ?

A way to program accelerators of all kinds from C++.

Solves three problems:

1. Enumerate devices (“accelerators” – regardless of vendor or type)
2. Share data with devices
3. Offload computation to devices

C++ with SYCL – Strong economic incentives

- Today: accelerators
  - NVIDIA does well by their customers, but…
    - Technology: Everything is Nvidia, and everything is a GPU
    - Legal: CUDA license is VERY strongly NVIDIA only
    - Economics: Proprietary is expensive
  - C++ with SYCL
    - Technology: Multivendor (not just Nvidia), *and* Multiarchitecture (not just GPUs)
    - Legal: Open
    - Economics: Highly competitive – lowest barriers to entry

C++ with SYCL – Strong technical incentives

A New Golden Age for Computer Architecture

High-level, domain-specific languages and architectures, freeing architects from the chains of proprietary instruction sets, along with demand from the public for improved security, will usher in a new golden age for computer architecture.

SOFTWARE CHALLENGE

Diverse and evolving workloads enable hardware innovation

Source: John L. Hennessy, David A. Patterson, Communications of the ACM
https://cacm.acm.org/magazines/2019/2/234352-a-new-golden-age-for-computer-architecture/fulltext

Today

- What/why is C++ with SYCL ?
- Learn C++ with SYCL
- Assignment: write a program using C++ with SYCL that distributes work across multiple devices
  - Resources: our book, free tools from Intel, free cloud resources from Intel
- Q&A

sycl::

It’s “just” C++ with extensions to:

1. Enumerate devices (“accelerators” – regardless of vendor or type)
2. Share data with devices
3. Offload computation to devices

Discuss: Why can’t C++ do this on its own?

sycl::

It's "just" C++ with extensions to:

1. Enumerate devices ("accelerators" – regardless of vendor or type)
2. Share data with devices
3. Offload computation to devices

Discuss: Why can't C++ do this on its own?

Single source – multiple targets

Disjoint memories

Heterogeneous compute

Enumerate devices

I really should say: find and connect to devices.

Why do I emphasize “enumerate”?

After we enumerate/find – we can connect.

Key concept: queue

- A queue is where we send requests to move data and/or offload computations
- A queue is permanently attached to a precise physical device
- We can have many queues
- A physical device can be pointed to by many queues

Share data with devices

SYCL has two methods, different but equally useful:

- buffers
- USM (unified shared memory)

We'll use buffers in our exercise.

Buffers are explicit. Helps correctness and runtime optimizations.

USM is easier for reusing C++ code. Allow for undisciplined accesses which may (often) destroy performance.

Not all devices can support USM, but the trend is for all devices to support.

Offload computation to devices

Kernels

Scaling happens when data parallelism is exploited.

Kernels are essentially “do this to one item”
and “parallel for” asks to do the kernel to all items.

Clear thinking is important. CUDA, OpenCL, SYCL – all are kernel oriented

Today

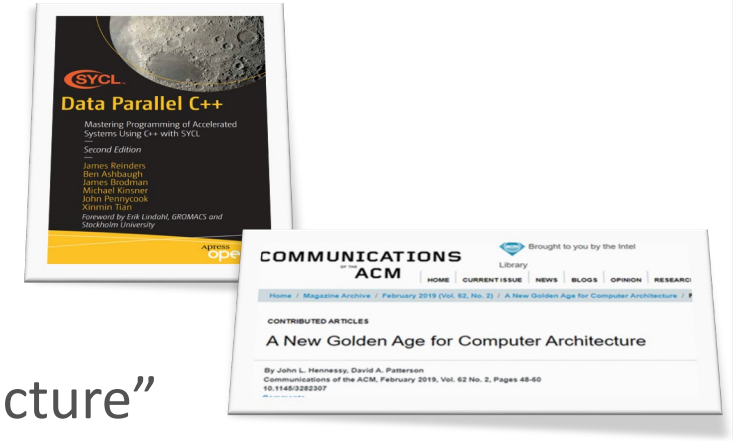
- What/why is C++ with SYCL ?
- Learn C++ with SYCL
- Assignment: write a program using C++ with SYCL that distributes work across multiple devices
  - Resources: our book, free tools from Intel, free cloud resources from Intel
- Q&A

Quickly (how much time is there?)

- Let’s peak at the code and logging in

https://github.com/jamesreinders/syclmultiple

Assignment



- Get Developer Cloud account (free)
- Login, download, run program
- From website – download copy of 2nd edition book
- From website – read the "New Golden Age of Computer Architecture"
- BIG PART:

  Modify the edge.cpp program to do more in parallel using multiple GPUs (or: use the CPU too)

  Extend the program any way you wish.

  Try other algorithms if you please.

  ****more suggestions on the website****
- By end-of-day Wednesday – email to me:
  - Short explanation of what you did (500 words or less)
  - Output from your program (describe in 100 words or less what it means)
  - Your source code (only files you changed or created)

Resources

- QR code goes to web-site
  - Github README.md points to web-site
- Web-site has links to resources
  - On-boarding instructions (sign-up, ssh)
  - Download and run edge.cpp
- Github has copy of this presentation
  - PDF
  - edge.cpp ; Makefile ; and more!
- Contact me
  - slow response e-mail ; Zoom chat and office hour ; class again next Thursday

https://github.com/jamesreinders/syclmultiple

Contact me!

See you all again next Thursday

Before then…

Email : james . r . reinders @ intel.com
(~once a day responses, less on weekends, be CLEAR and COMPLETE)
please put "Cornell Class" as the first words in your subject

“Office hour” on Monday – HIGHLY OPTIONAL! Includes a 24/7 Team Chat.

Time: Monday Sep 18, 2023 4pm Eastern Time (US and Canada)
ZOOM CALL: https://tinyurl.com/2023CornellZoom
Meeting ID: 852 9477 8720
Passcode: 130235

Today

- What/why is C++ with SYCL ?
- Learn C++ with SYCL
- Assignment: write a program using C++ with SYCL that distributes work across multiple devices
  - Resources: our book, free tools from Intel, free cloud resources from Intel
- Q&A
  - With resources QR-code / URL

Q&A

https://github.com/jamesreinders/syclmultiple


intel

Next time – possible topics (subject to change)

- Discuss results
- Q&A
- Amdahl – how much parallelism is there
- 1024 chickens?
- Ponder the future
  - What does it mean to program in the future?
  - Mechanics vs. Users?
- Q&A

Musings (rough for now)

- What does it mean to program in the future?
- Mechanics vs. Users?

- AI is really about programming differently – letting the computer do MORE of the work

- ASM -> Fortran -> spreadsheets -> AI