IWOCL 2026

SYCL: An Update from the Khronos Working Group

Thomas Applencourt, Argonne National Laboratory

on behalf of the SYCL WG

We have a Rock Too!

General Update of Current Ecosystem

- SYCL upstreaming, effort by Intel, to LLVM (`liboffload`)
- Qualcomm is working on a SYCL implementation for their SoC
- Lots of activity in AdaptiveCpp, oneAPI and UXL Foundation
- SYCL WG contracted Kitware to add official SYCL support to CMake

As usual, a little advertisement:

- As usual, everything is public, join us at `https://github.com/KhronosGroup/SYCL-Docs`
- You can also join the WG, and assist in our weekly call[1]

[1] Not that expensive for academic, and you can push for feature you want, or don't want...

SYCL 2020: Now

- We are now in revision 11
- Only bug fixes / clarifications go into the main specification
- New features are going via KHRonos extensions

We now have 8 KHRs:

1. *sycl_khr_default_context*
2. *sycl_khr_queue_empty_query*
3. *sycl_khr_group_interface*
4. *sycl_khr_max_work_group_queries*
5. *sycl_khr_queue_flush*
6. *sycl_khr_work_item_queries*
7. *sycl_khr_static_addrspace_cast*
8. *sycl_khr_dynamic_addrspace_cast*

SYCL 2020: Future

Have 3 KHR extensions in the pipeline[2]:

- **Compile Time Property**: Better code gen and errors for property list
  - *https://github.com/KhronosGroup/SYCL-Docs/pull/980*
- **Free launch**: No event created by default
- **Better Work-Group Memory Allocation**: Replacement for local accessor

And then we'll stop SYCL 2020, and move to SYCL Next.

[2] Final name tbd

SYCL Next

Current Plan: Simplify the Spec for Both Users and Implementers.

We want some feedback! The spec is there to reflect usage and user need

- Some KHRs will be merged to main spec (default context, max work group queries , etc) based on user feedback and usage.
- Take the opportunity to deprecate features, or other change.

SYCL Next: Deprecation and Other Change. Let's discuss

Possibility to deprecate some API:

- Buffer Interface: Lot of corner cases, hard to implement, high overhead, too verbose, low adoption
- Image: Not implemented by implementer
- Vec: We have *`marray`*
- Specialization Constant: Low usage, and no consensus on API/design

Other miscellaneous possible change:

- Make in order queue the default
- Add an memcopy 2d without buffer
- Pushing the idea of default context to his natural conclusion (*`sycl_free`* taking only a pointer, etc)

To finish: SYCL Common Repo for Utilities?

People have nice SYCL libraries:

- I have a nice ”free-standing” C++23 ***println***-like for SYCL, for example[3]
- People may want to write a ”buffer-like” API replacement

Should we have a ”boost” like for SYCL? Where? UXL? Just a new GitHub org?

[3] *https://github.com/TApplencourt/sycl-print.git*

Conclusion

- We have new KHR extensions, please use them
- We have a plan for SYCL Next, thanks for your feedback
- Don’t hesitate to open issue, PR!
- Thanks a lot!