Writing elegant host-side code with modern-C++ wrappers for the CUDA runtime API

Eyal Rozenberg
DB Architectures Group
CWI Amsterdam

CWI

Are you in the right lab?

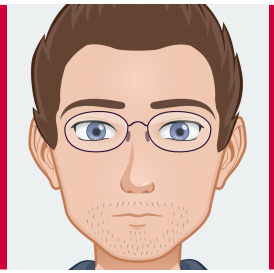
We are going to:

- Work on rewriting a very simple CUDA program:
  - gradually transitioning to a nicer API
  - keeping the original functionality
- Write some **host**-side code

  ... not device-side code (= no kernels)
- Use some C++11 and C++14

  … but don't panic if you're not an expert :-)
- If we make good time, we might dive into the innards at the end.

About me



- Moving back and forth between Academia and Industry
- Wrapping up a post-doc stint with the CWI DBMS architecture group
  - Home of MonetDB – a FOSS analytics-oriented column store DBMS
- Focused on Analytic DBMS Architecture, and how GPUs should affect it
- Have done some work on lightweight compression

  … look for **`libgiddy`** on GitHub
- Modern C++ enthusiast

  … the language is getting more expressive, a lot less contortion necessary

(My) motivation

- CUDA kernel code is adopting C++11, C++14 constructs
- Host-side code isn't going in that direction *at all* (!)
- In fact, we are stuck with a purely C-style API:
  - Explicit error checking everywhere (no exceptions)
  - Need to pass addresses for pre-allocated results
  - A bazillion different functions with long names
  - Lots of pointers and No RAII (Resource Acquisition is Initialization)
- I found myself "wrapping up the mess" with nicer-looking code

  … which eventually covered much of the Runtime API

Let's get to it...!

- Start your virtual machines
- You should have a file named `lab.tar.gz` (somewhere).
- Execute `tar xvf lab.tar.gz && cd lab.`
- Execute `make clone; make v01.`
- Edit the file `vectorAdd.cu`
  - You can use nsight as a visual editor
  - … and there's also vi, emacs and possibly others
- Let me know you're set up and ready
  by taking your hands off the keyboard.

(Hands-on segment)

The API wrapper library

- It's already released:

`https://github.com/eyalroz/cuda-api-wrappers`

- **Licensing**: Permissive – (3-clause) BSD license
- **Users**: Occasional individual forkers and a couple of projects

  … not yet in production systems
- **Documentation**: Mostly-complete, doxygen-based
- **Adoption by nVIDIA?**: Not yet, pester them about it!
- **Support**: By yours truly; use the GitHub issues page.

Thank you
and happy(er) coding.