Lost in Translation: Challenges in Automating CUDA to OpenCL Translation

Paul Sathre, Mark Gardner, Wu-Chun Feng

Depts. of Computer Science and IT

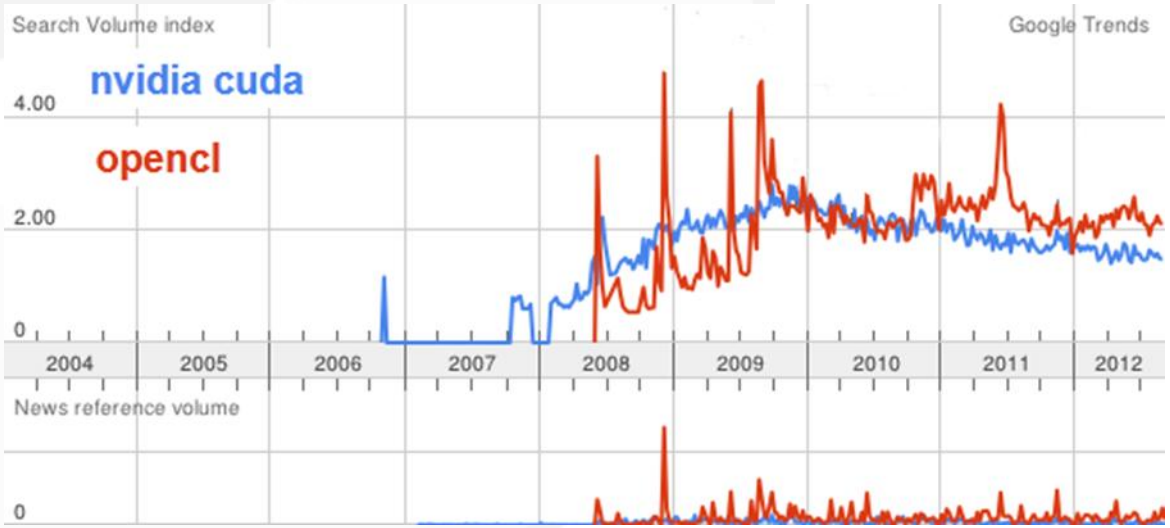
P2S2 Workshop @ ICPP – 2012.09.10

VirginiaTech
Invent the Future

SyNeRG
synergy.cs.vt.edu

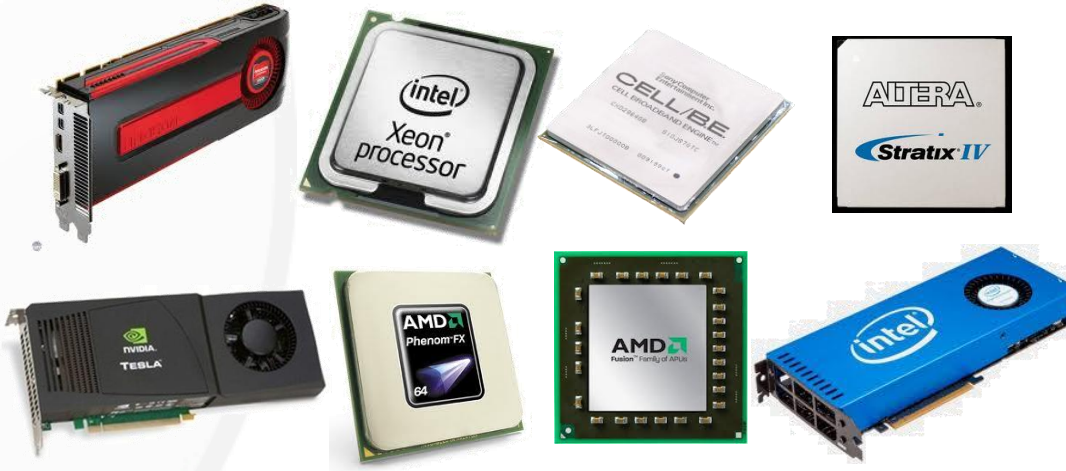
CUDA in HPC

5	National Supercomputing Center in Tianjin China	**Tianhe-1A** - NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050 NUDT
6	DOE/SC/Oak Ridge National Laboratory United States	**Jaguar** - Cray XK6, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA 2090 Cray Inc.
10	National Supercomputing Centre in Shenzhen (NSCS) China	**Nebulae** - Dawning TC3600 Blade System, Xeon X5650 6C 2.66GHz, Infiniband QDR, NVIDIA 2050 Dawning



www.top500.org, www.google.com

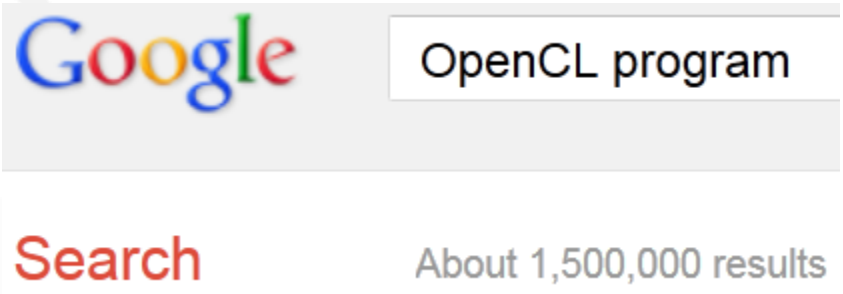
Supported Platforms

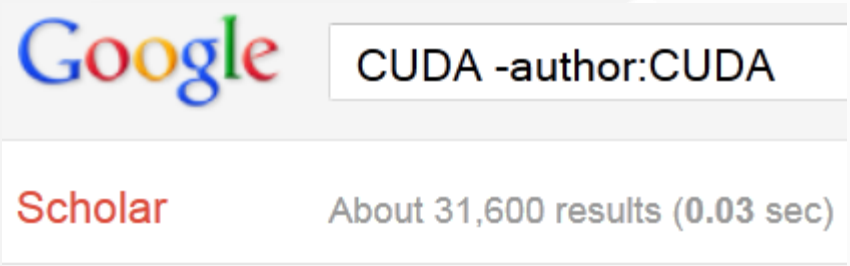


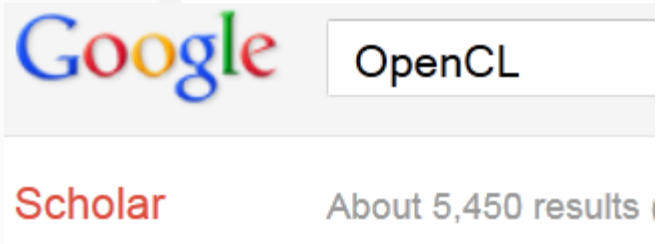
Future platforms too!

www.google.com

Why CUDA to OpenCL?







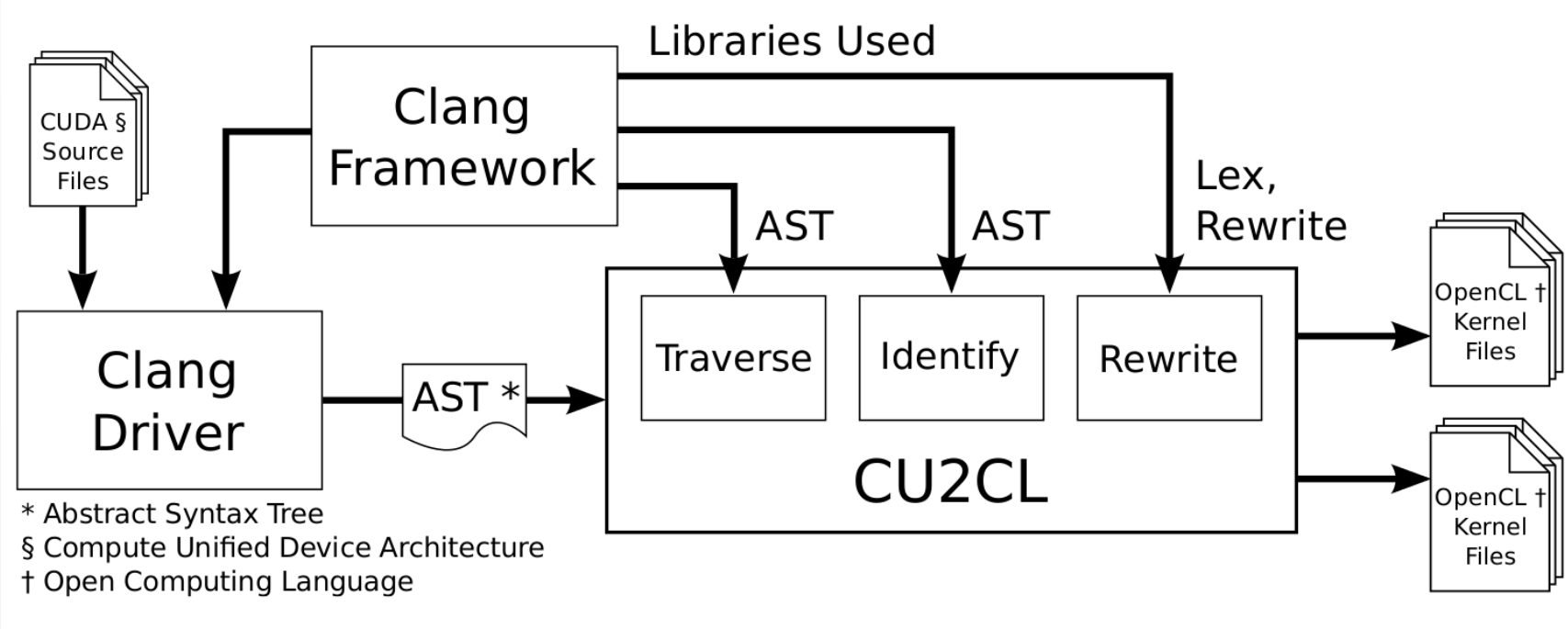
- Leverage existing investment
- Enable future development in OpenCL
- Extensive model similarities
- Hand translation is tedious and error-prone

www.google.com

Lost in Translation

- Why CUDA to OpenCL?
- **Our approach to translation**
- **Profiling 90+ CUDA applications**
- Key Insights
  - Limitations of propagation
  - Source-level limitations
  - Important Subtleties
- Summary and Q&A

Our Approach: *CU2CL* (CUDA-to-OpenCL)

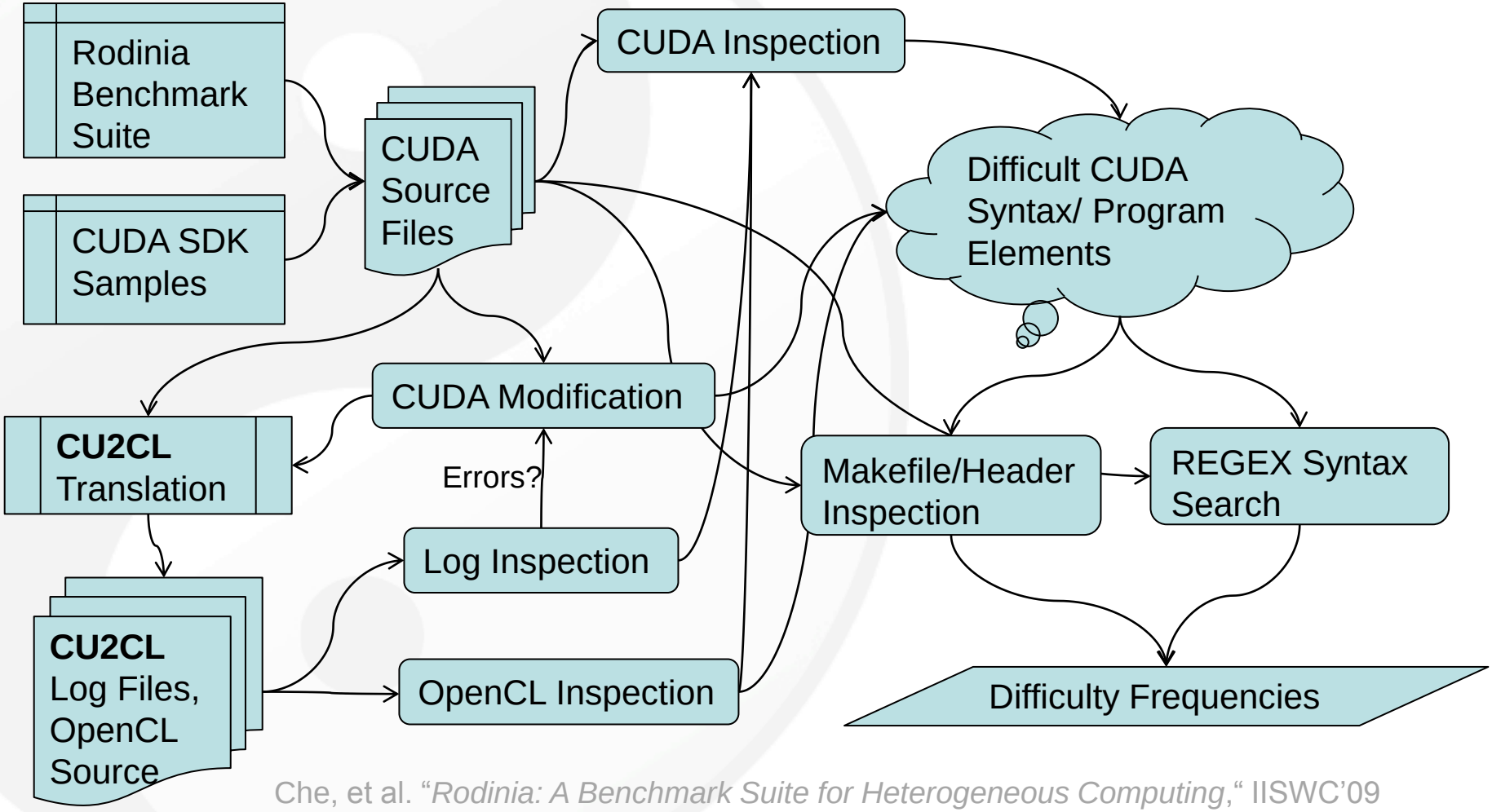



```
__pow( x[threadIdx.x], y[threadIdx.y])
```

```
native_pow( x[get_local_id(0)], y[get_local_id(1)])
```

Martinez, Gardener, and Feng, "*CU2CL: A CUDA-to-OpenCL Translator for Multi- and Many-Core Architectures*," ICPADS '11

Profiling 90+ CUDA Applications



Che, et al. "*Rodinia: A Benchmark Suite for Heterogeneous Computing*," IISWC'09
Che, et al, *"A Characterization of the Rodinia Benchmark Suite with Comparison to Contemporary CMP Workloads,"* IISWC'10

Lost in Translation

- Why CUDA to OpenCL?
- Our approach to translation
- Profiling 90+ CUDA applications
- **Key Insights**
  - **Limitations of propagation**
  - **Source-level limitations**
  - **Important Subtleties**
- Summary and Q&A

Limitations of Propagation

- CUDA and OpenCL are both separately compiled and linked
- Both depend on and modify a global state
  - CUDA's largely implicit, OpenCL's completely explicit
- But, translator cannot "see" outside file
  - How to infer state?
  - Ex. JIT Kernel Compilation
- Other rewrites affected too
  - Ex. Device Buffers
- How to guarantee completeness of translation?

Limits to Source Translation

- Functional portability restricted to high level source
- How to deal with non-source components?
  - CUDA Libraries? CUBLAST, CUFFT, CURAND, etc.
  - PTX and cubins?

- Device language barrier – OO Constructs
  - C++ (CUDA) vs. C99 (OpenCL)
  - Cannot directly implement device-side function templates

Device Initialization

NONE!

OR..

```
cudaSetDevice(int devID);
```

```
clGetPlatformIDs(…);
clGetDeviceIDs(…);
clCreateContext(…);
clCreateCommandQueue(…);
```

AND

```
//iterate over devices
clReleaseCommandQueue(…);
clReleaseContext(…);
clCreateContext(…, devices[devID]);
clCreateCommandQueue(..);
```

Kernel Argument Specification

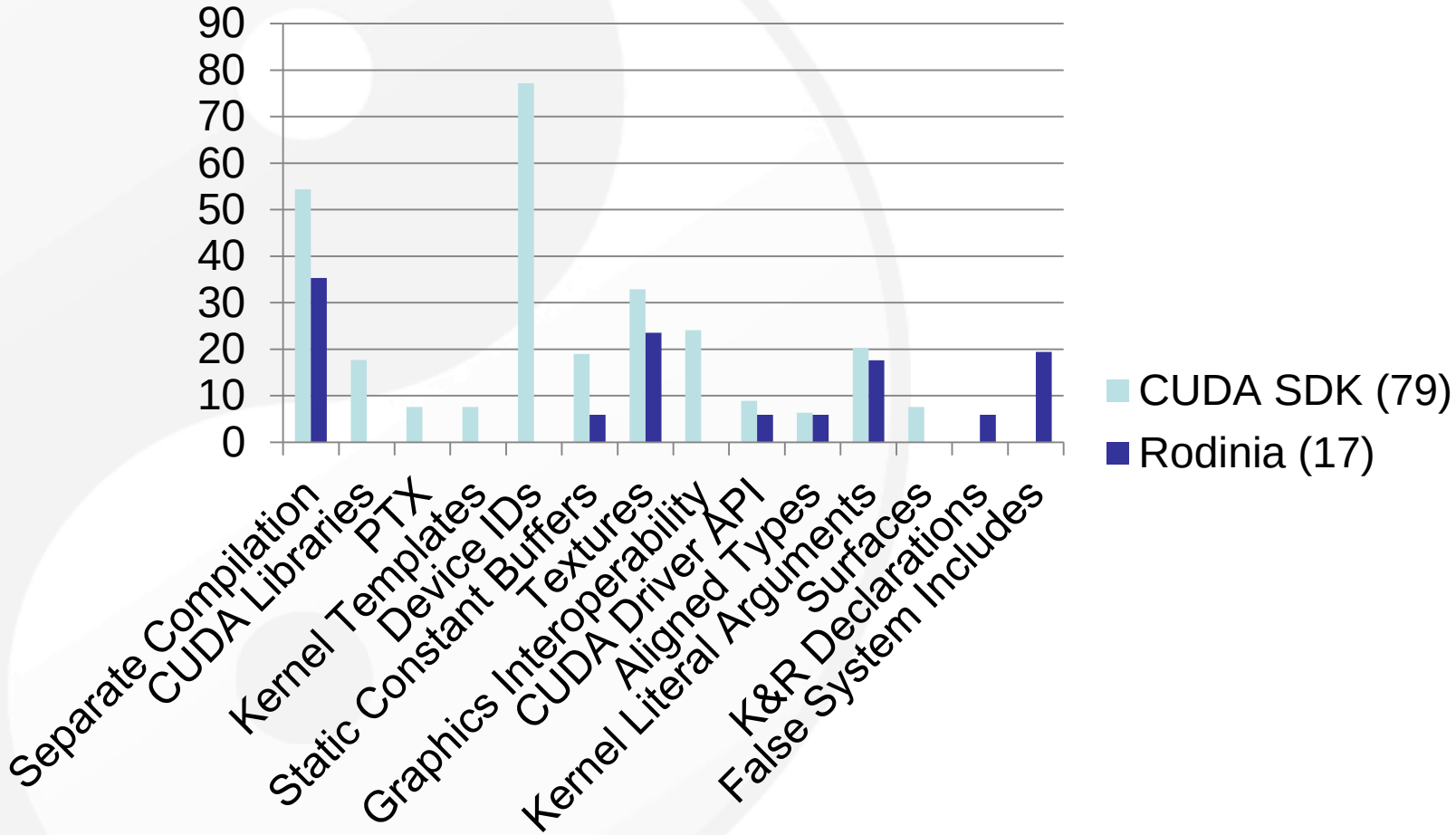
```
fooKern<<<grid, threads>>>(d_fooBuff, fooInt,
3.14f, MIN(bigInt, smallInt));
```

```
clSetKernelArg(fooKern, 0, sizeof(cl_mem), &d_fooBuff);
clSetKernelArg(fooKern, 1, sizeof(cl_int), &foo_int);
clSetKernelArg(fooKern, 2, sizeof(cl_float), &3.14f);
clSetKernelArg(fooKern, 3, sizeof(cl_int), &MIN);
clEnqueueNDRangeKernel(queue, fooKern, … );
```

```
cl_float tempPi = 3.14f;
cl_int tempMin = MIN(bigInt, smallInt);
clSetKernelArg(fooKern, 0, sizeof(cl_mem), &d_fooBuff);
clSetKernelArg(fooKern, 1, sizeof(cl_int), &foo_int);
clSetKernelArg(fooKern, 2, sizeof(cl_float), &tempPi);
clSetKernelArg(fooKern, 3, sizeof(cl_int), &tempMin);
clEnqueueNDRangeKernel(queue, fooKern, … );
```

Summary and Q&A

- Need translators to enable functional portability
  - Correctness first, then performance!
- Separation of compilation and linking complicates cross-file translations
- Source-level translators must actively recognize even what they can't handle
- .. and must be mindful of corner cases which require special handling

This material is based upon work supported by the National Science Foundation under Grant No. 0916719. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Percentage of Affected Applications