CU2CL

Automated Source-to-Source Translation from CUDA to OpenCL for General Purpose Accelerated Computing

Synergy/SEEC Fall Symposium 2015

Paul Sathre

VirginiaTech
Invent the Future

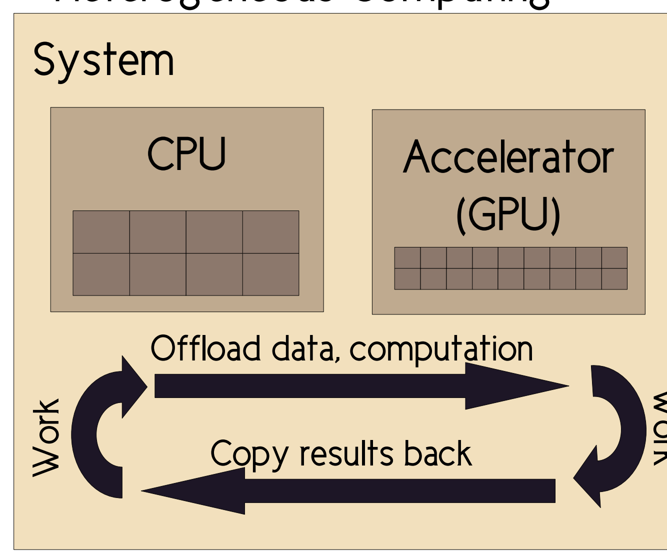
SyNeRG
synergy.cs.vt.edu

Parallel and Heterogeneous Computing is *Everywhere*

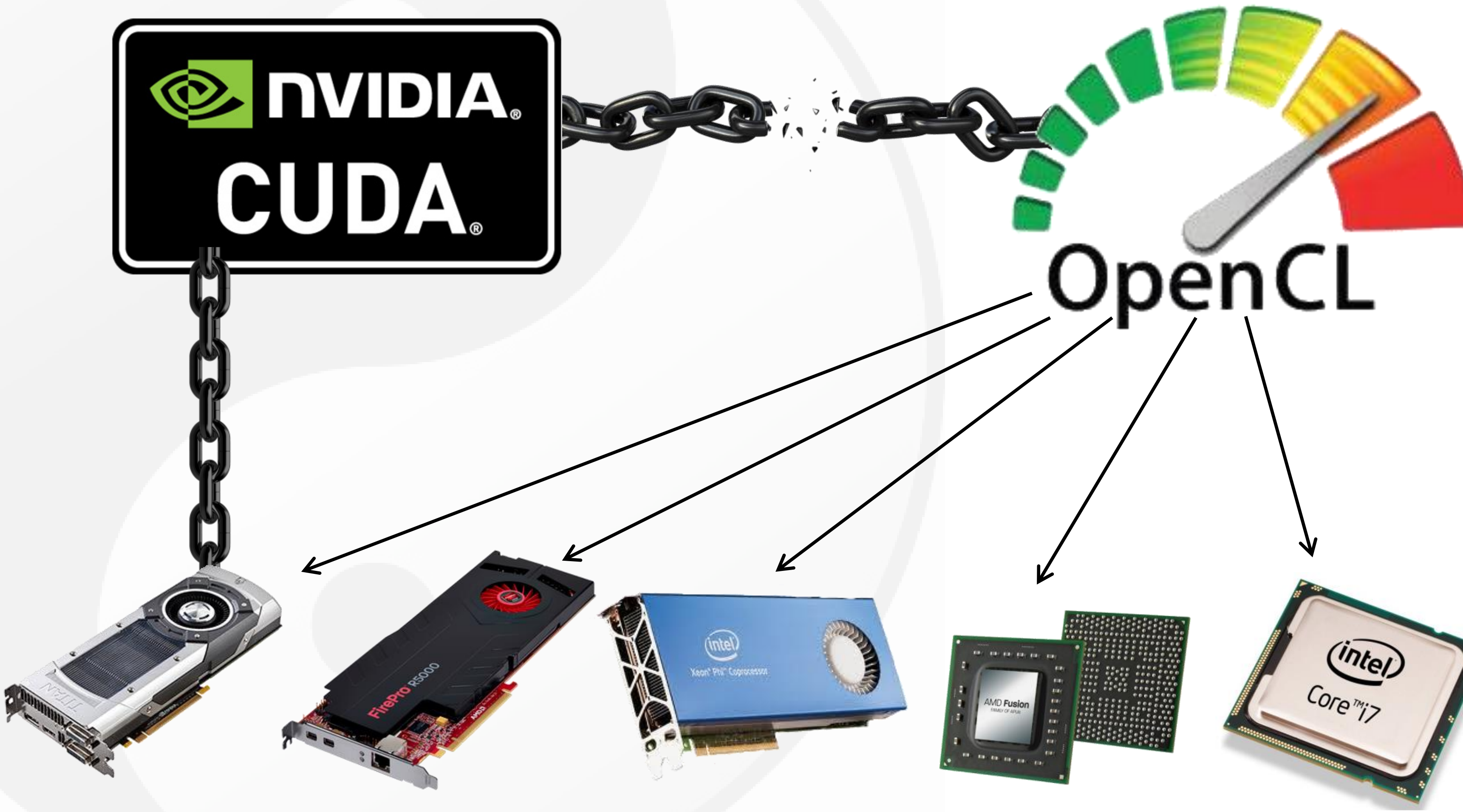



- Supercomputers to cell phones
- Mac OS X (since Snow Leopard)
  - *GPU-accelerated*
- Adobe Photoshop CS6
  - *GPU-accelerated*

Heterogeneous Computing



Functional Portability



Translation Is Easy …

- … when there is NO ambiguity in the translation between languages

- High-level language ⟶ low-level representation, e.g., C ⟶ LLVM

```
x * y + z →
%tmp = mul i32 %x, %y
%tmp2 = add i32 %tmp, %z
```

- Between languages, e.g., CUDA ⟶ OpenCL

```
__powf(x[threadIdx.x], y[threadIdx.y]) →
native_pow(x[get_local_id(0)], y[get_local_id(1)])
```

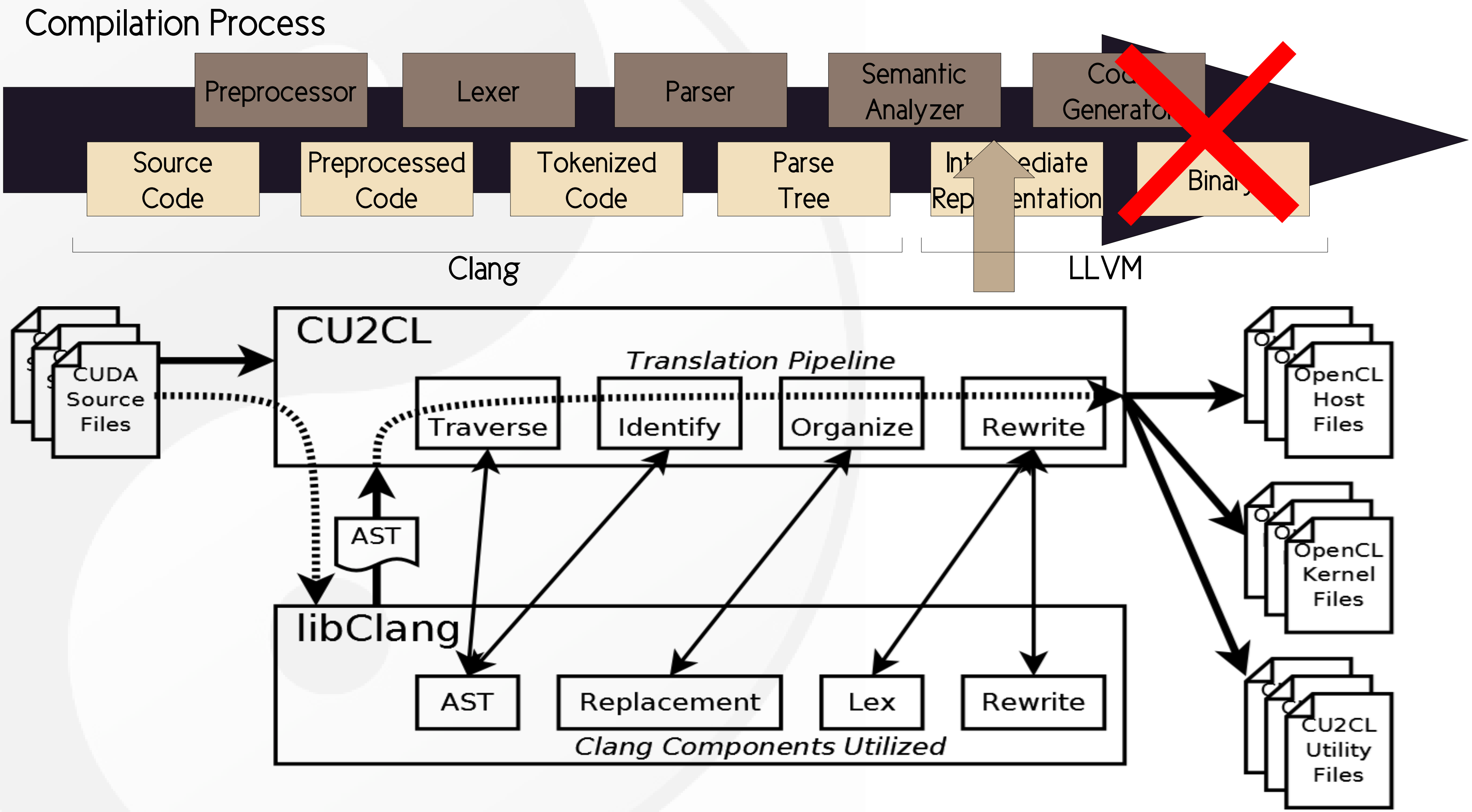
Translation isn't easy ...

- ... when there IS ambiguity (or lack of one-to-one mapping) in the translation between languages

- Idiomatic Expressions
  - *"Putting all your eggs in one basket"* → ??
  - *CUDA* **threadfence()** → ??

- Dialects
  - *Latin American Spanish vs. Castilian Spanish → English*
  - *CUDA Runtime API vs. CUDA Driver API → OpenCL*

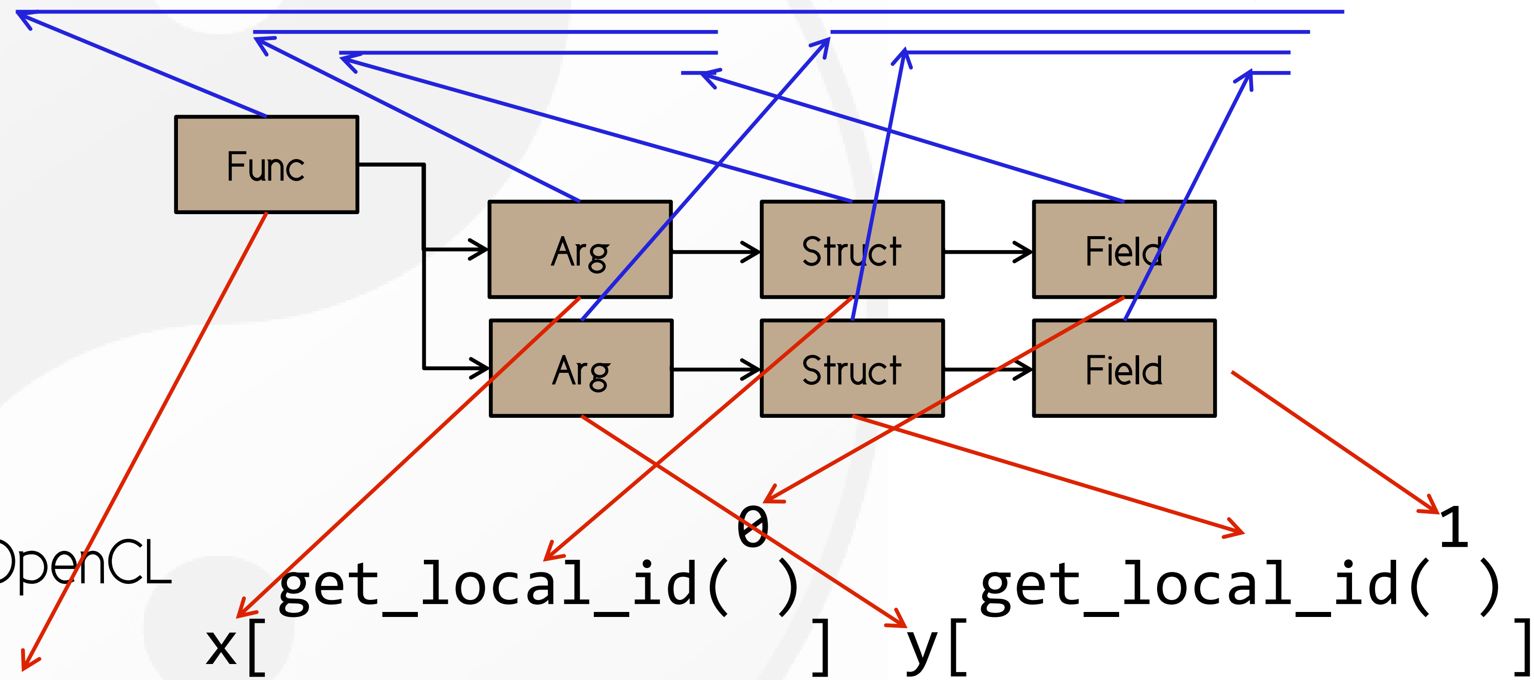
CU2CL Translator Prototype



String-based Recursive Rewriting

- CUDA

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



- OpenCL

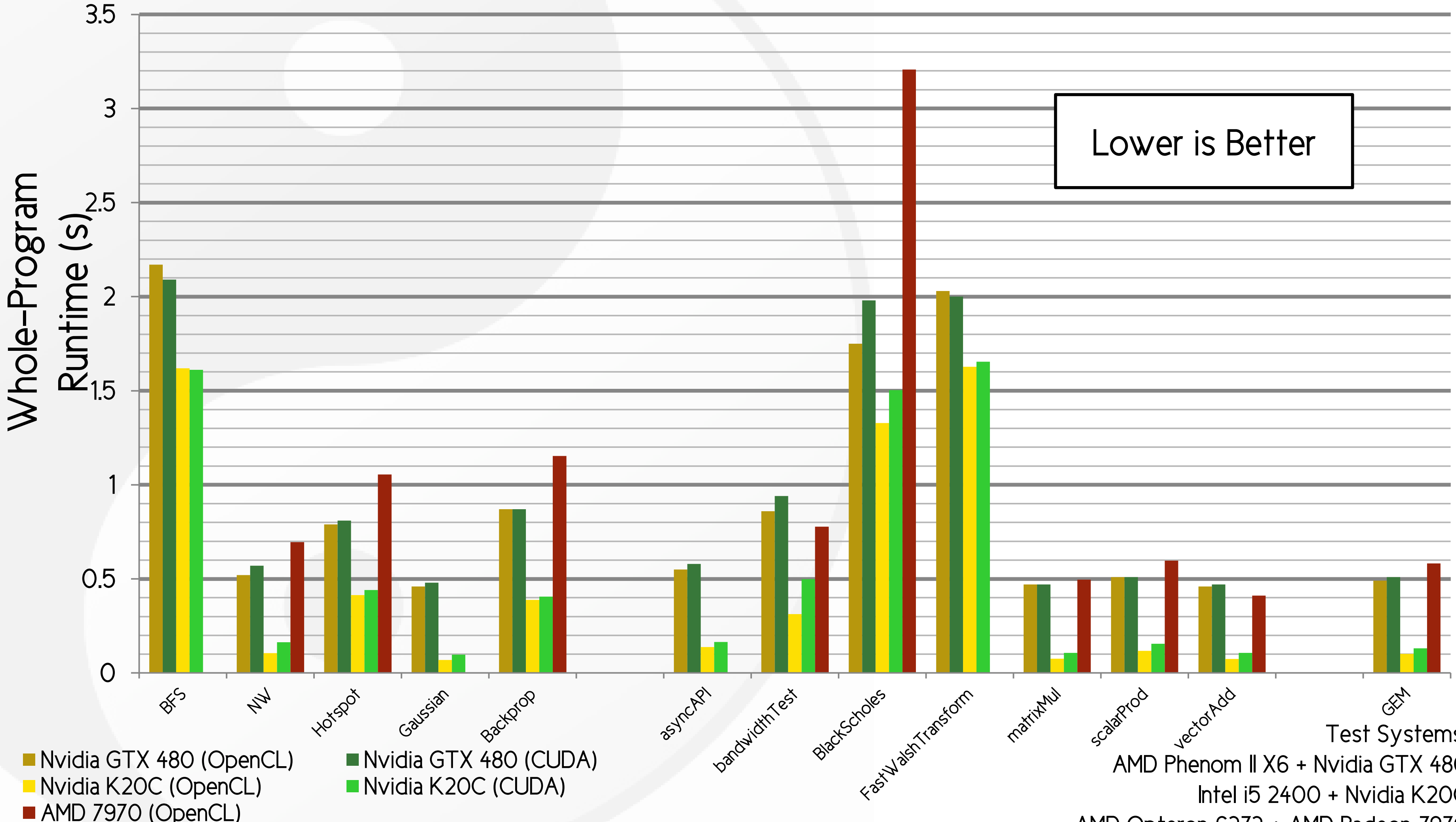
```
native_pow(x[get_local_id(0)], y[get_local_id(1)])
```

```
<sourcefile>:<byteoffset>:<origlength>:“native_pow(x[get_local_id(0)], y[get_local_id(1)])”
```

Translator Coverage

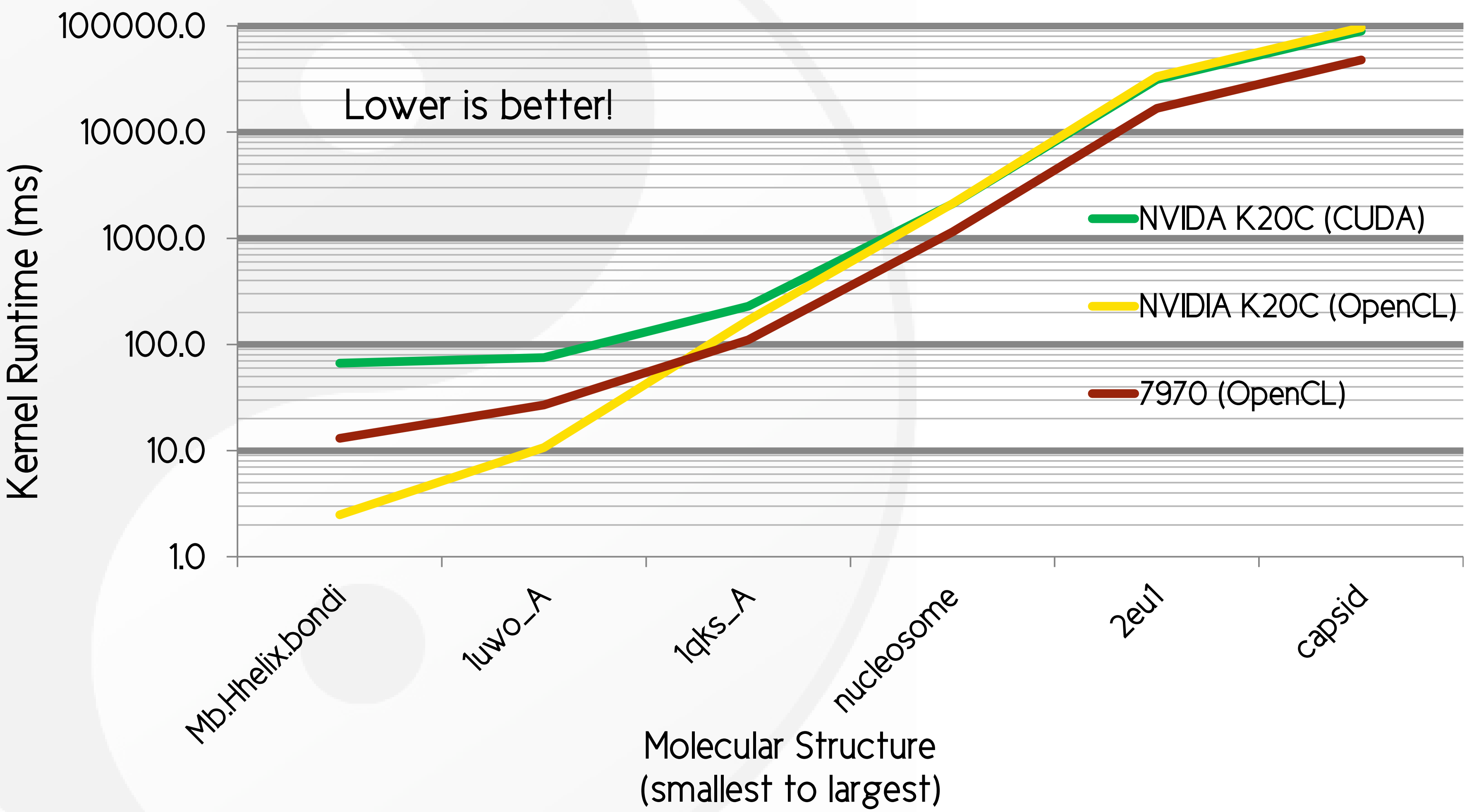
	Application	CUDA Lines	OpenCL Lines Changed	Percent Automatically Translated	Dwarf
SDK Samples	*asyncAPI*	*135*	*5*	*96.3*	–
SDK Samples	*bandwidthTest*	*891*	*5*	*98.9*	–
SDK Samples	*BlackScholes*	*347*	*14*	*96.0*	*Dense Linear Algebra*
SDK Samples	*FastWalshTransform*	*327*	*30*	*90.8*	*Spectral Methods*
SDK Samples	*matrixMul*	*351*	*9*	*97.4*	*Dense Linear Algebra*
SDK Samples	*scalarProd*	*251*	*18*	*92.8*	*Sparse/Dense Linear Algebra*
SDK Samples	*vectorAdd*	*147*	*0*	*100*	*Sparse/Dense Linear Algebra*
Rodinia	*Back Propagation*	*313*	*24*	*92.3*	*Dynamic Programming*
Rodinia	*Breadth-First Search*	*306*	*35*	*88.6*	*Graph Traversal*
Rodinia	*Gaussian*	*390*	*26*	*93.3*	*Dense Linear Algebra*
Rodinia	*Hotspot*	*328*	*2*	*99.4*	*Structured Grids*
Rodinia	*Needleman-Wunsch*	*430*	*3*	*99.3*	*Dynamic Programming*
	Fen Zi	*17768*	*1786*	*89.9*	–
	GEM	*524*	*15*	*97.1*	*N-body Methods*
	IZ PS	*8402*	*166*	*98.0*	–

Translated Application Performance (Runtime Overhead)



Test Systems:
AMD Phenom II X6 + Nvidia GTX 480
Intel i5 2400 + Nvidia K20C
AMD Opteron 6272 + AMD Radeon 7970

Translated Application Performance: GEM



Future Efforts

- Modularize the frontend/backend
  - Separate CUDA structure identification from OpenCL string generation, so that potentially other languages can be "plugged in"
- De-C++ kernel code
  - De-templating
  - De-classing
- Altera/Non-standard CL mode
  - Generate offline compile scripts (could be useful for HSAIL)
  - Modify whatever boilerplate is required.
- Expanded CUDA support
  - Textures
  - Error handling
  - Dynamic Shared Memory
  - other API components

Related Work

- **Swan:** wrapper API around either CUDA OR OpenCL host API with a Basic CUDA-to-OpenCL Kernel Translator
  - M. J. Harvey and G. D. Fabritiis, "Swan: A tool for porting CUDA programs to OpenCL," Computer Physics Communications, vol. 182, no. 4, pp. 1093–1099, 2011.
- **MCUDA:** CUDA → OpenMP translator
  - J. A. Stratton, S. S. Stone, and W. W. Hwu, Languages and Compilers for Parallel Computing. Springer-Verlag, 2008, ch. MCUDA: An Efficient Implementation of CUDA Kernels for Multi-core CPUs, pp. 16–30.
- **Ocelot / Caracal**: PTX → LLVM IR / AMD CAL IR
  - G. F. Diamos, A. R. Kerr, S. Yalamanchili, and N. Clark, "Ocelot: A Dynamic Optimization Framework for Bulk-Synchronous Applications in Heterogeneous Systems," in 19th International Conference on Parallel Architectures and Compilation Techniques, 2010, pp. 353–364.
  - R. Dominguez, D. Schaa, and D. Kaeli, "Caracal: Dynamic Translation of Runtime Environments for GPUs," in 4th Workshop on General Purpose Processing on Graphics Processing Units , 2011, pp. 5:1–5:7.
- **CUDAtoOpenCL**: MS Project at UIUC: CUDA-to-OpenCL translator (AFAIK, never released, and now defunct)
  - D. Nandakumar, "Automatic Translation of CUDA to OpenCL and Comparison of Performance Optimizations on GPUs," 2011. {Online}. Available: http://hdl.handle.net/2142/24279

Contributions

- Automatically translate upwards of 80-90% of most apps!
  - perfect translation in a few cases
- Maintainable code, retained formatting and comments
  - Translation in place maintains structure
- Collaboration and PR with AMD via translation of open source Fluidsv3 smooth particle hydrodynamics Visualization/Simulation
- Binary and source freely available for non-commercial use
- CHREC funders have had new 0.7.0b version since December
  - Unified translation of multi-file codebases

Conclusions

- Robust automatic source translation from CUDA to OpenCL is (mostly) achievable
  - *Not straightforward, but moving towards full functional portability*
  - *Some important (current) limitations*
    - » Libraries, Device C++, NVIDIA-specific functions/ hardware behaviors
    - » Maturation of OpenCL ecosystem may help (i.e. SYCL)

- Once translated, application performance is retained
  - *Work to utilize new devices is reduced*
  - *Software is able to reach a broader audience*

- What used to take weeks/months/years to do by hand, now takes seconds!

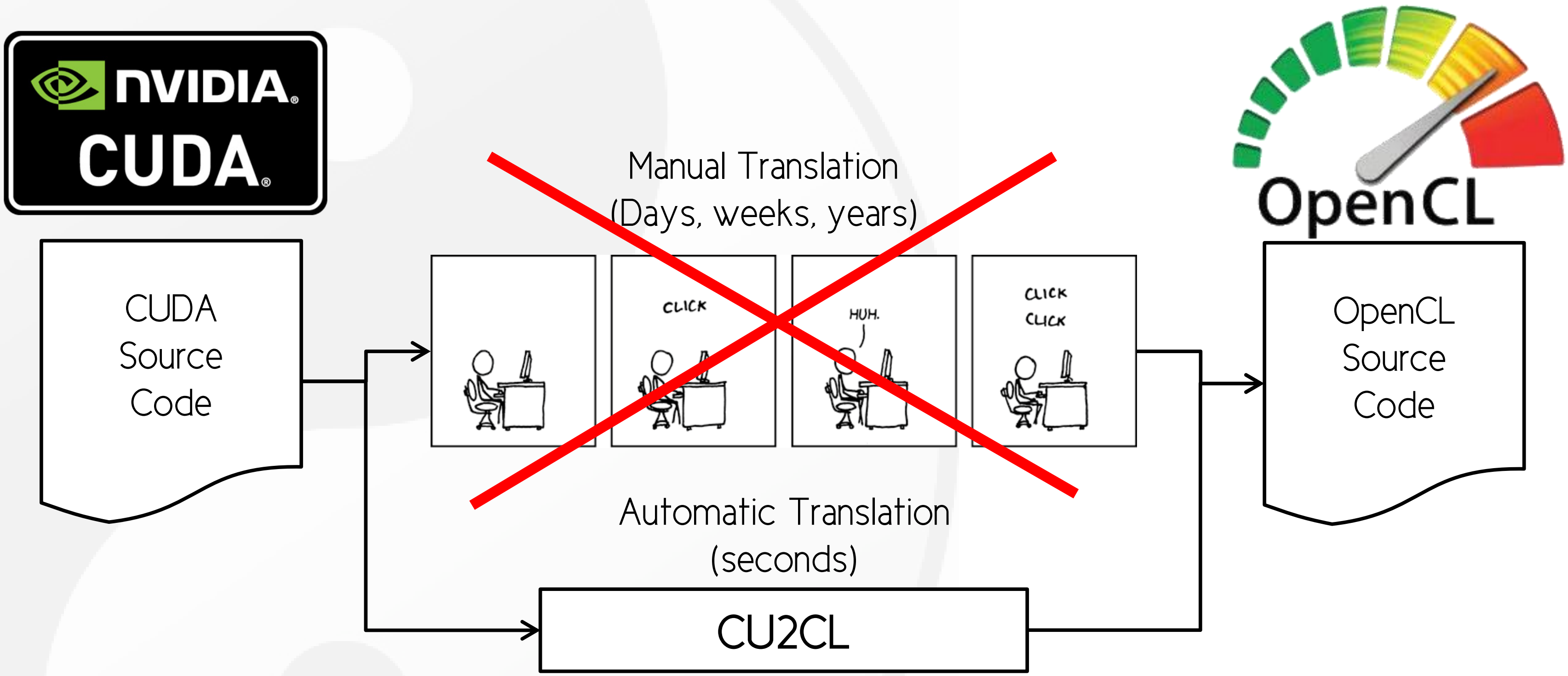
Acknowledgements

- Advisors: Dr. Wu-chun Feng, Dr. Mark Gardner
- Alumni: Gabriel Martinez, Paul Sathre
- This work was supported in part by NSF I/UCRC IIP-0804155 via the NSF Center for High-Performance Reconfigurable Computing (CHREC).

NSF

CHREC
NSF Center for High-Performance Reconfigurable Computing

Questions?



Linux build and tool scripts available: http://chrec.cs.vt.edu/cu2cl

Source release available: https://github.com/vtsynergy/CU2CL

xkcd.com

Appendix

Globalized Translations

- Reimplemented plugin as *Clang Tool*
  - Can run consecutive translator instances on all source files making up a binary within a single invocation

- Intelligent cache and merge of rewritten code via *Replacement* objects

```
<sourcefile>:<byteoffset>:<origlength>:<replacementtext>
fooKern.cu:1337:38:"native_pow(x[get_local_id(0)], y[get_local_id(1)])"
```

- Deferred OpenCL boilerplate generation
  - partial initialization and deconstruction code generated per file
  - global initializers/deconstructors generated after last source file processed

Complex Semantic Conversions

Device Identification

– *CUDA uses* int*, OpenCL uses opaque* cl_device

– *To change devices in CUDA, use* cudaSetDevice(int id)

– *To change devices in OpenCL, use…*

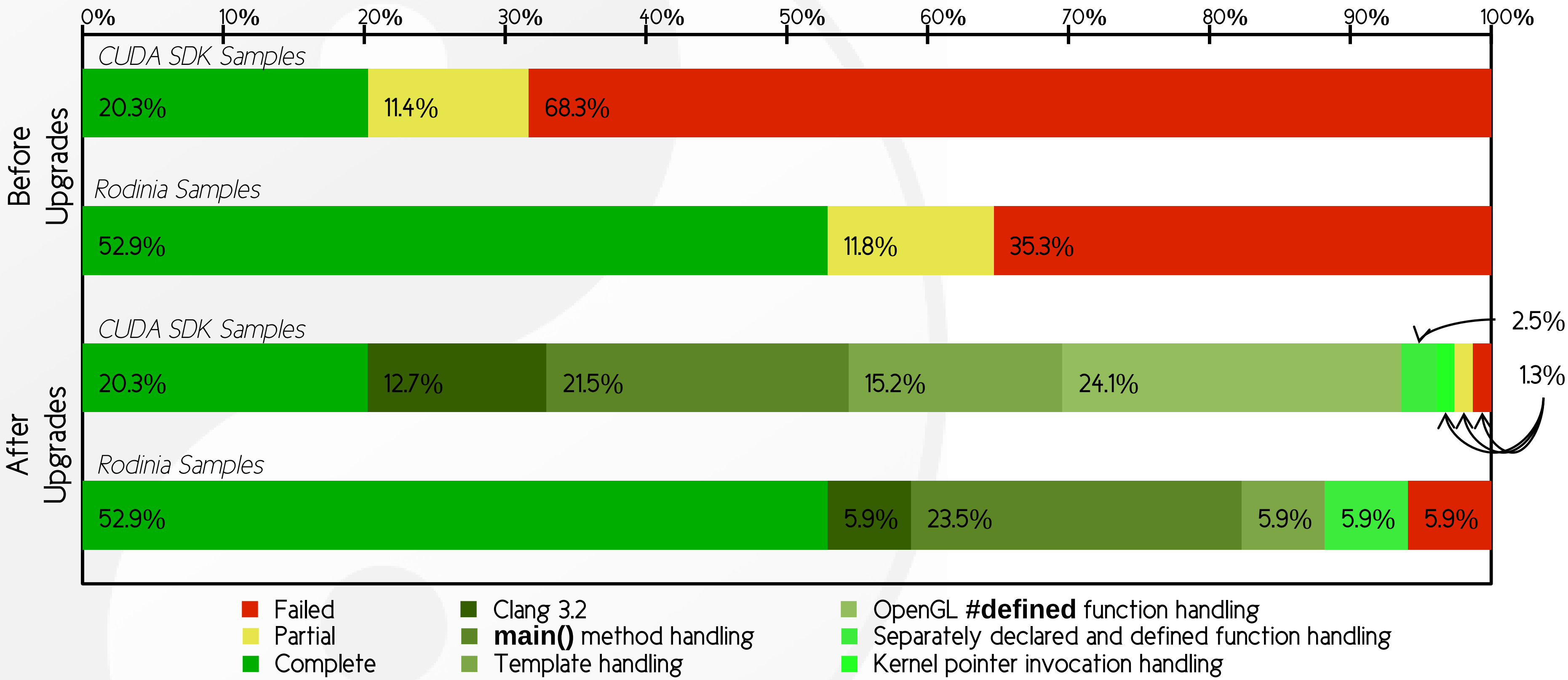
```
//scan all devices
//save old platform, device, context, queue, program, & kernels
myDevice = allDevices[id]
ClGetDeviceInfo(...);      //get new device's platform
myContext = clCreateContext(...);
myQueue = clCreateCommandQueue(...);

//load program source
clBuildProgram(...);
myKernel = clCreateKernel(...);
```

– *Implement our own handler to emulate and encapsulate*

```
        /*CU2CL Warning -- CU2CL Identified cudaSetDevice usage*/
__cu2cl_SetDevice(devID);
```

CU2CL Reliability



AMD collaboration

Fluids v.3: Open Source Windows+CUDA fluid simulation, translated to OpenCL w/ CU2CL and run on AMD W9100s, demo at ISC'14 in Leipzig