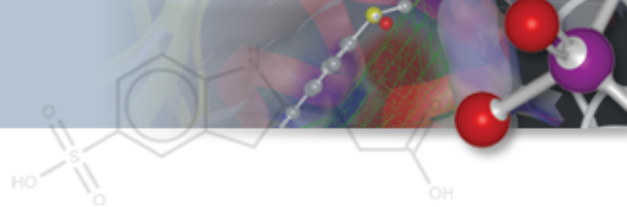
OpenEye
Scientific Software

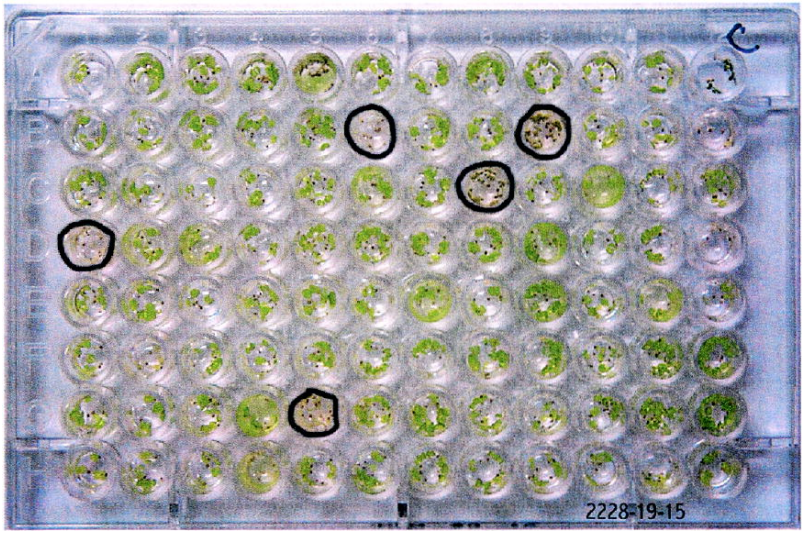


FastROCS
An OpenCL Deployment Story

Brian Cole
Lead Technology Strategist &
Toolkit Project Manager

(Virtual) Screening

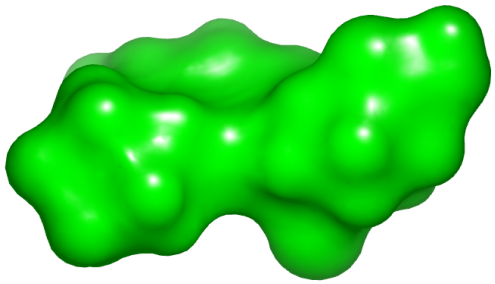
- Searching for the next miracle drug



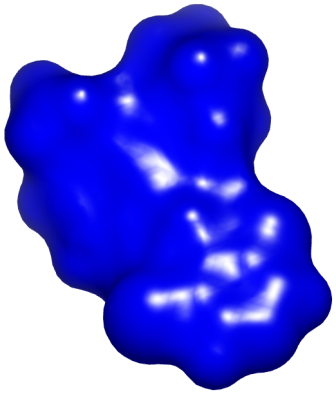
- ~10^{10} compounds
  - 1 well = $1
- "Drug-like" chemical space
  - ~10^{33}

What is ROCS?

Rapid Overlay of Chemical Structures: High speed ligand-based database searching tool

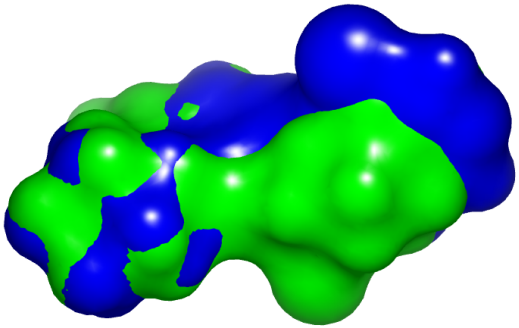


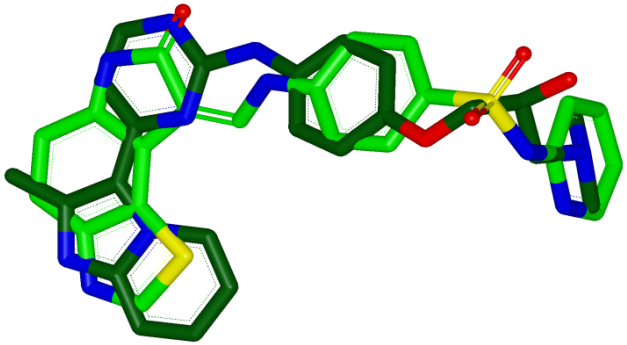
Query



Database

Align and score by shape and chemistry

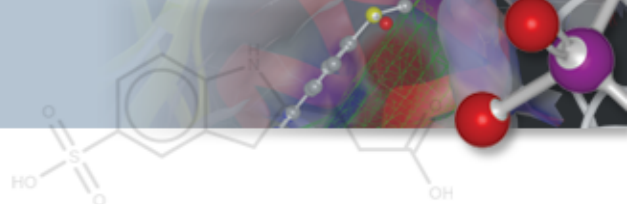




Embarrassingly parallel over the database

ROCS user base

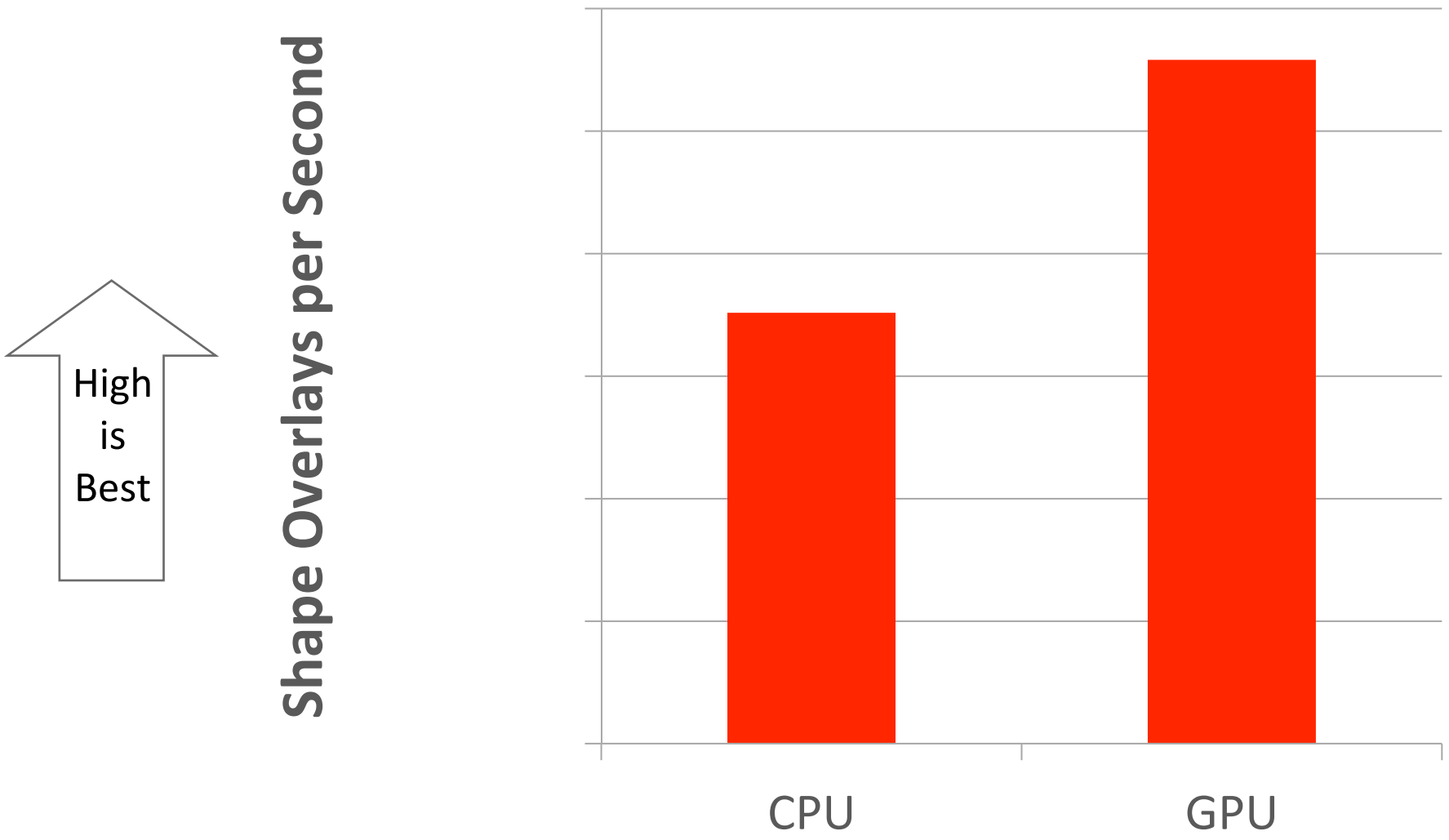
- Every Pharma R&D
- Many BioTechs
- Many Universities
- National Labs and Research Centers
- Other software companies



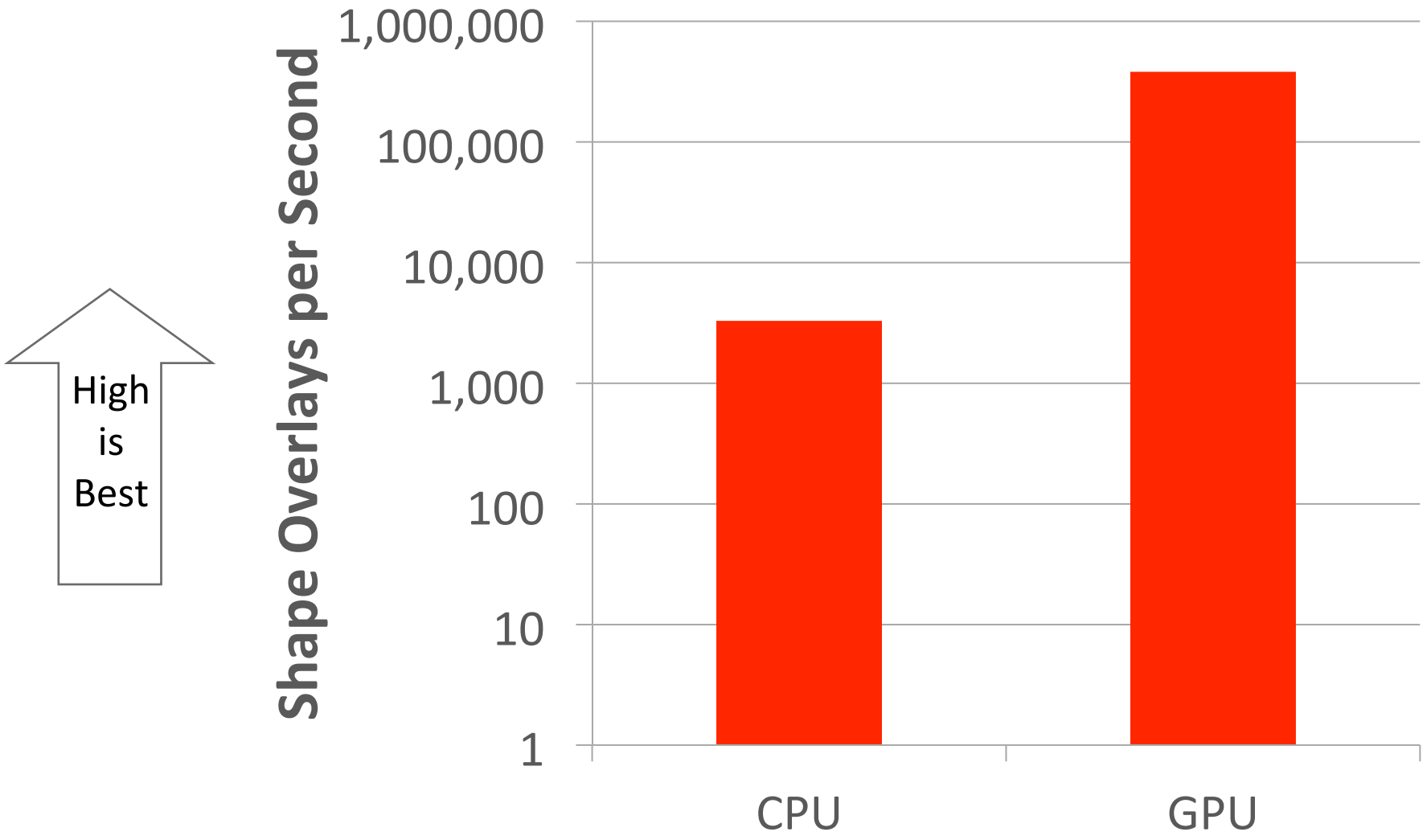
Fast ROCS

Fast Rapid Overlay of Chemical Structures

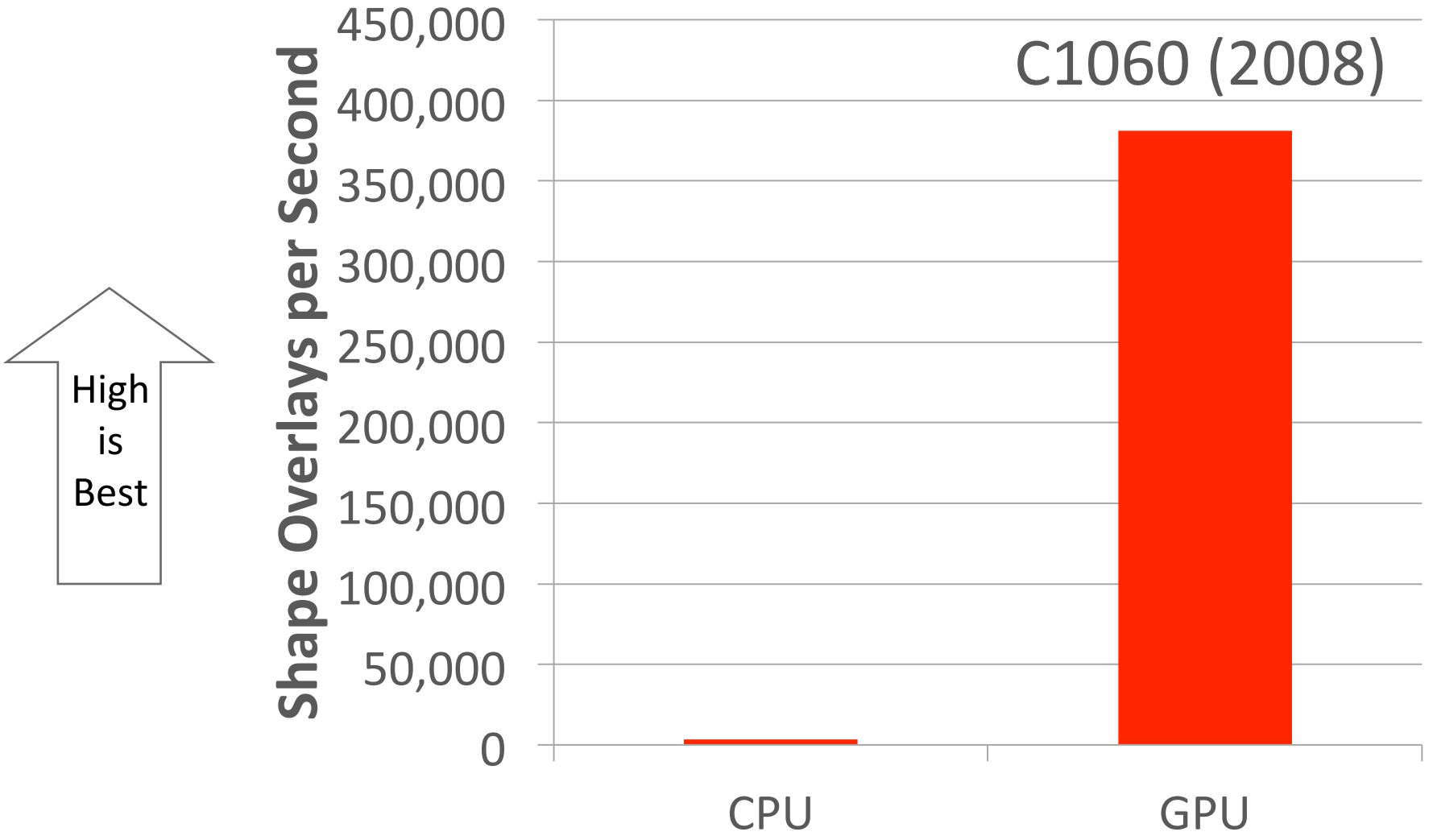
What is FastROCS?

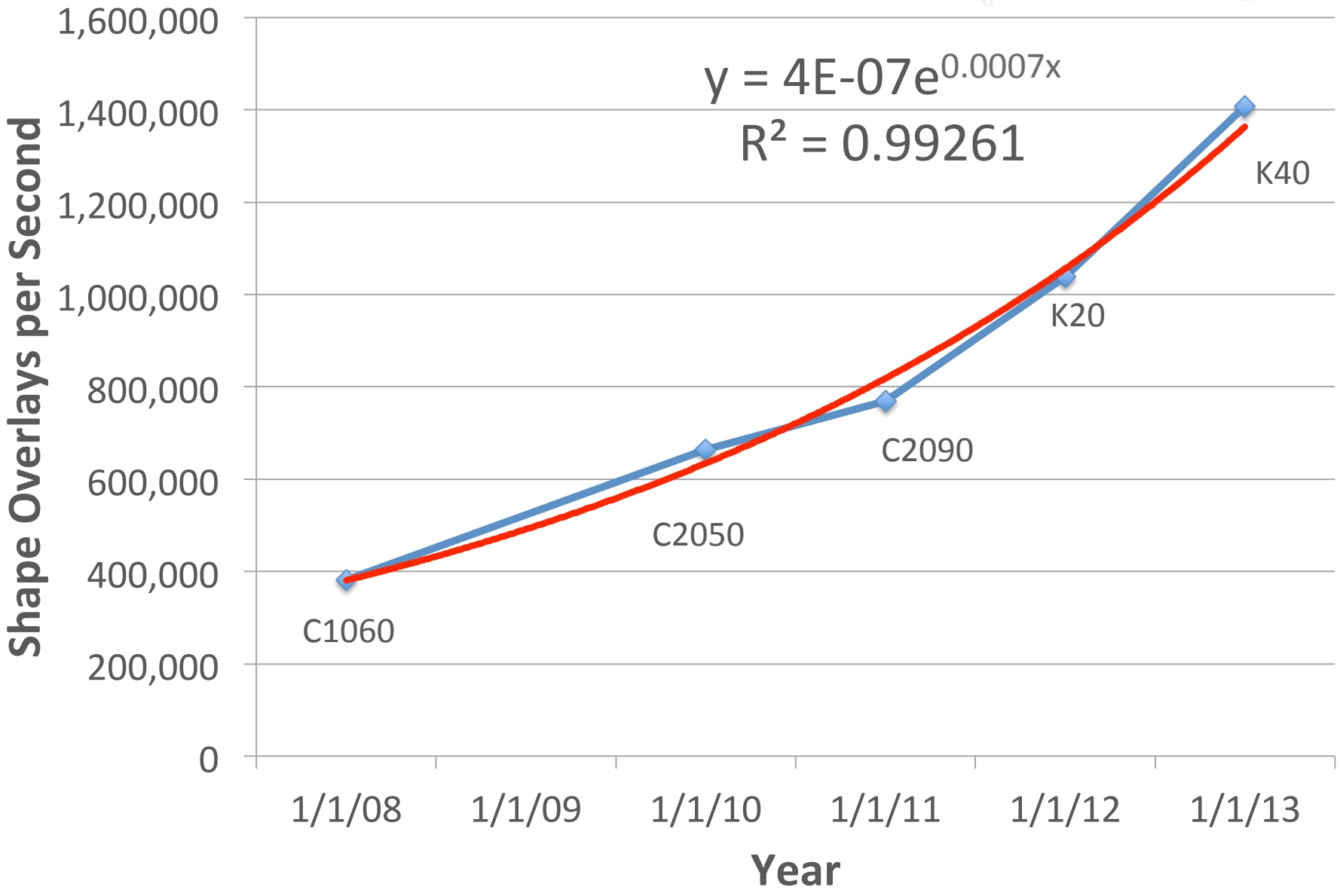


What is FastROCS?



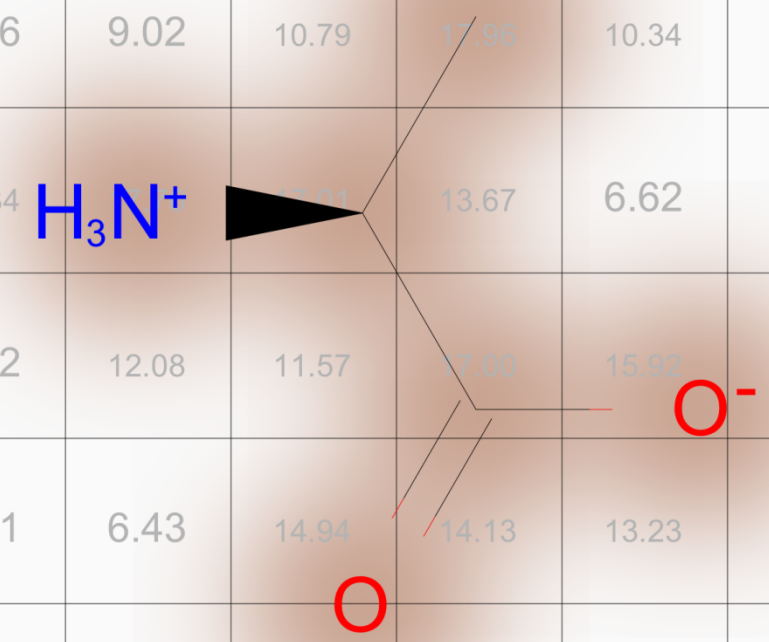
What is FastROCS?





$O(n) < O(n^2)$

0.02	0.22	1.01	1.59	6.70	11.15	6.42	1.28	0.09	0.00
0.10	1.27	5.76	9.02	10.79	17.96	10.34	2.06	0.24	0.02
0.19	2.50	11.34	17.91	13.67	6.62	5.66	1.67	0.17	
0.13	1.70	7.72	12.08	11.57	17.00	15.92	13.61	4.02	0.41
0.03	0.40	1.81	6.43	14.94	14.13	13.23	11.31	3.34	0.34
0.00	0.05	0.97	6.52	15.16	12.18	3.80	3.25	0.96	0.10



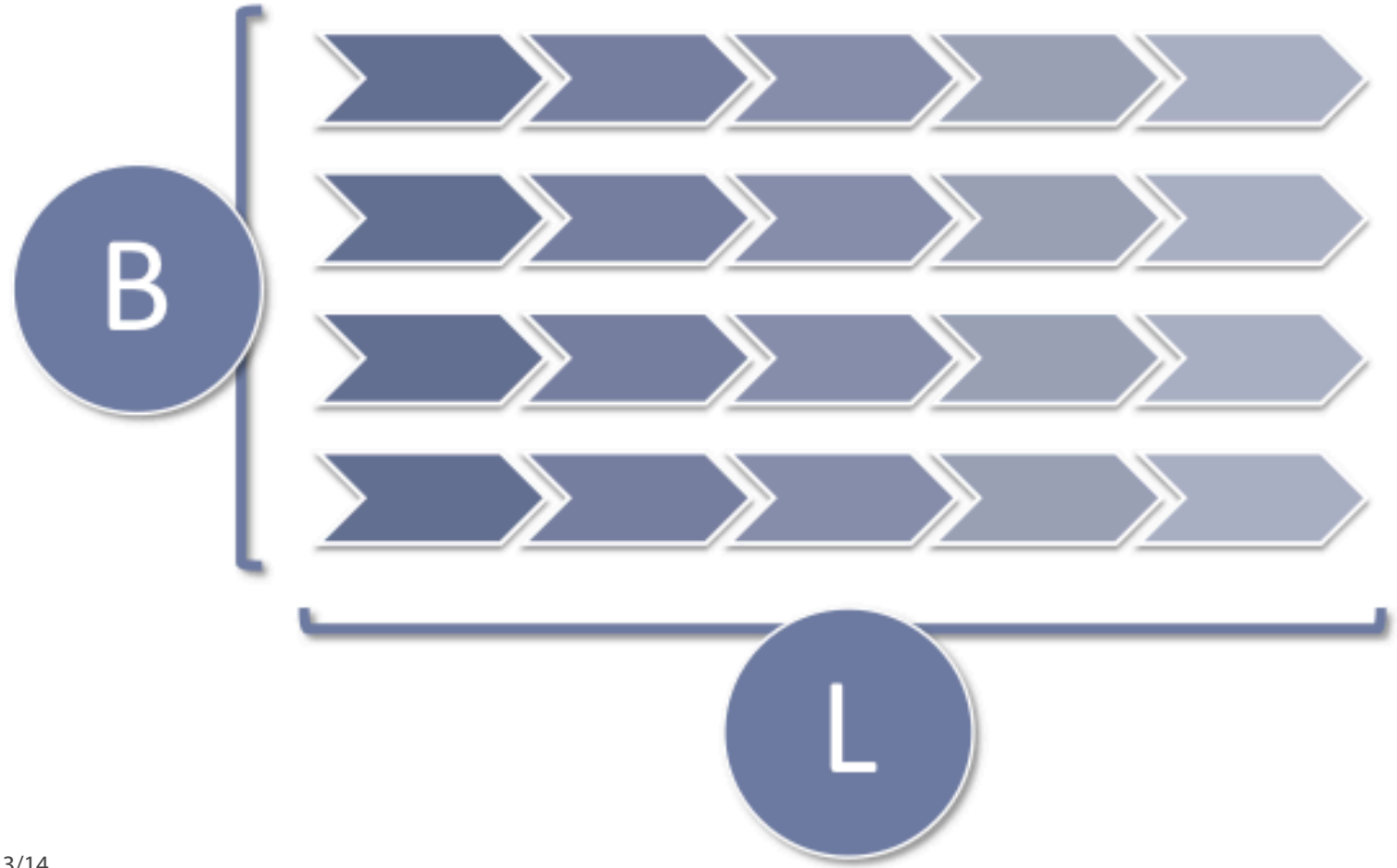
Then just “clever use of game mechanics”

Single kernel: 679 lines of OpenCL

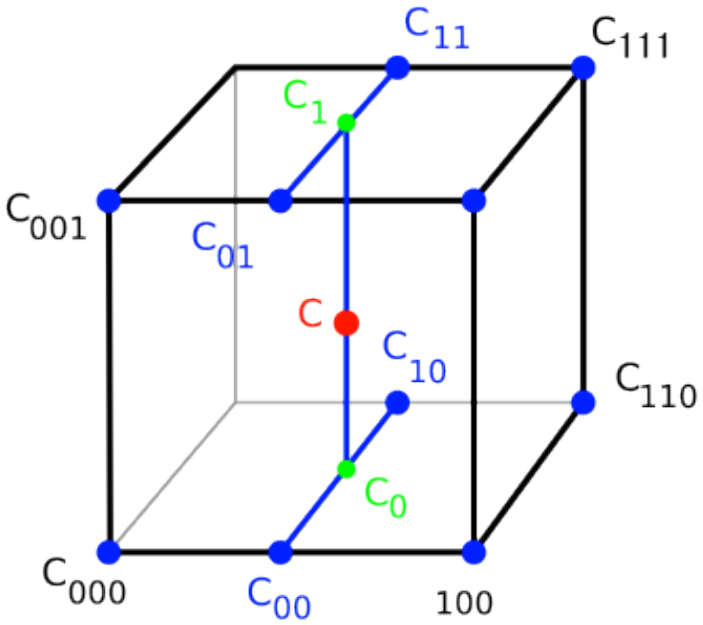
- A highly tuned optimizer per work item
- Working set stored in __local memory
  - async_work_group_copy
- Objective function bounded by read_imagef
  - image3d_t
    - CL_HALF_FLOAT
  - sampler_t
    - CLK_FILTER_LINEAR
- 64 registers per work item
  - High occupancy used to hide texture latency

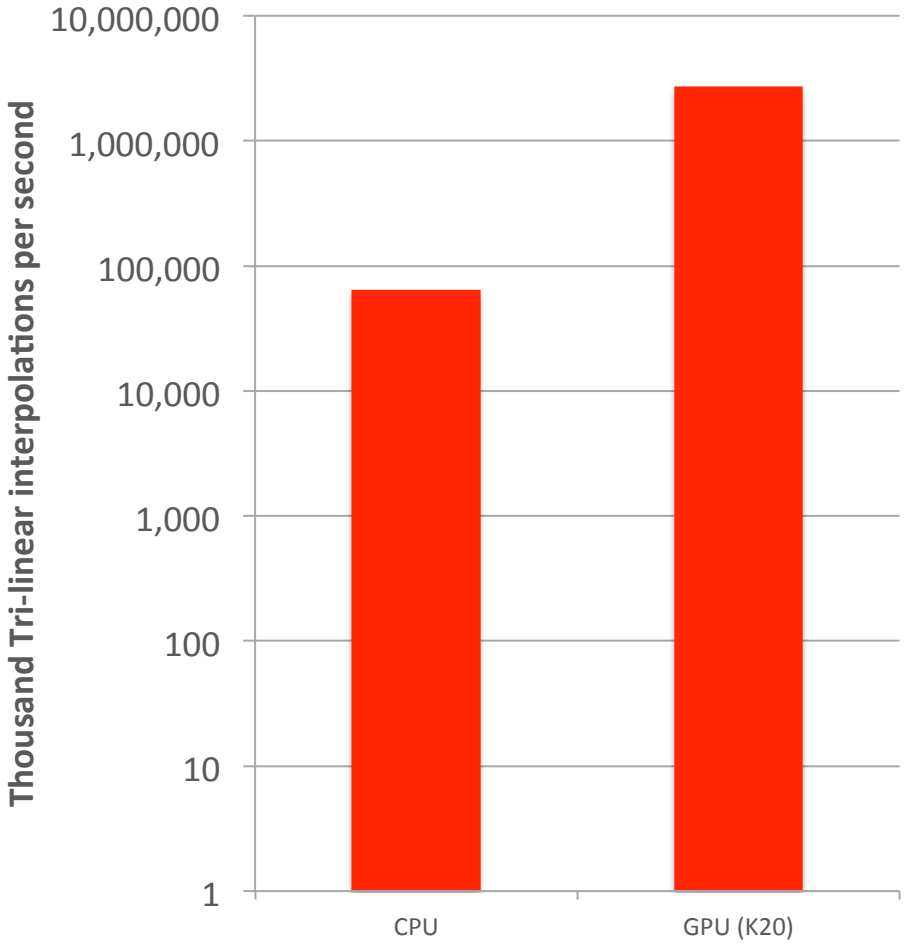
Little's Law: Embarrassingly parallel's best friend

Bandwidth × Latency = Concurrency



Tri-linear interpolation throughput

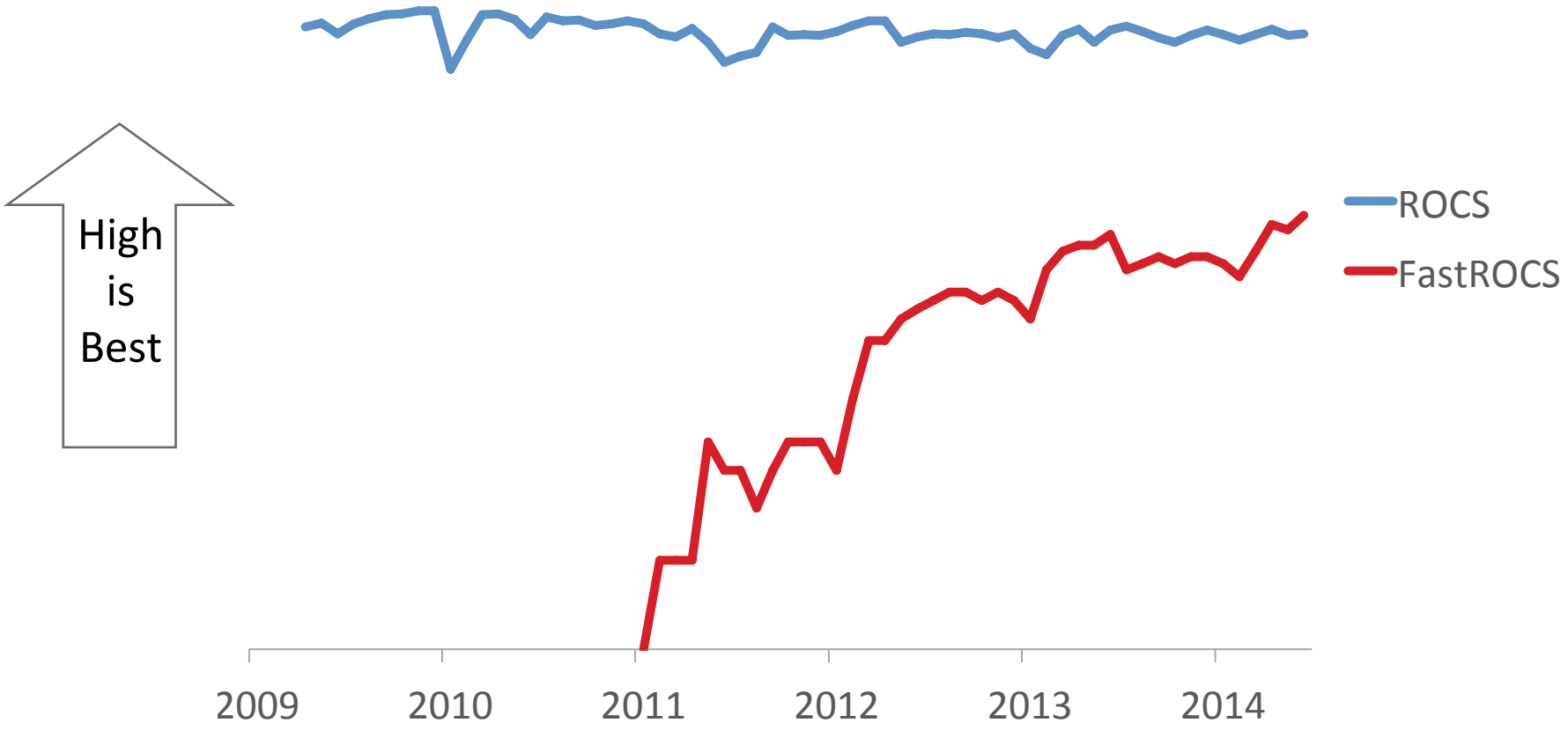




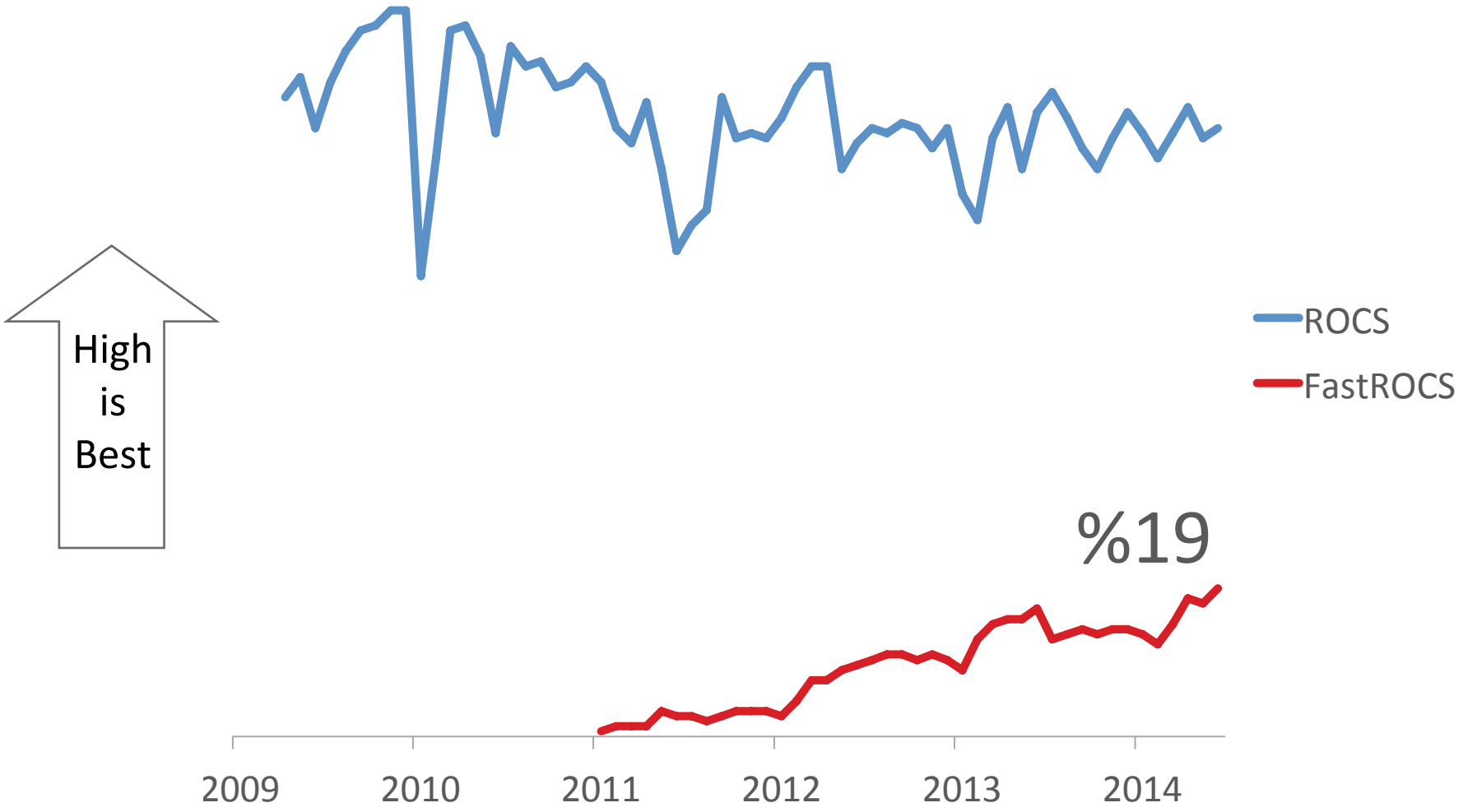
ROCS user base

- Every Pharma R&D
- Many BioTechs
- Many Universities
- National Labs and Research Centers
- Other software companies

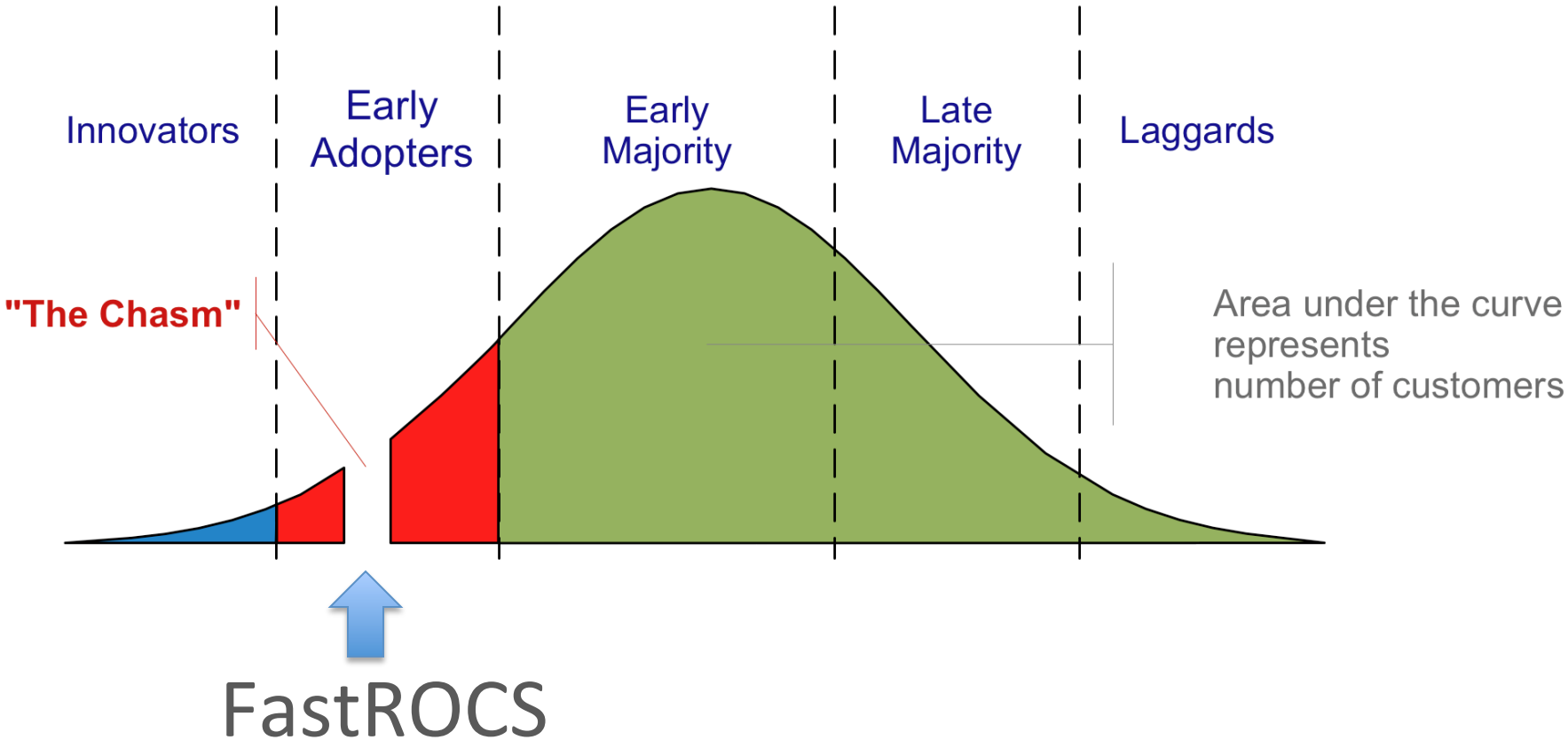
Licenses by month



Licenses by month (linear scale)



Technology Adoption Lifecycle



What's in the "chasm"?

- "GPUs are hard to deploy"
- "GPUs are hard to maintain"
- "The results aren't bitwise comparable"
- "There's nothing else to run on the GPU"

ROCS quick start guide

- tar –xzf ROCS-3.1.1-RHEL5-x64.tar.gz
- openeye/bin/rocs query.sdf database.oeb.gz

FastROCS quick start

- crtl-alt-F1 (to switch to a non X-server terminal)
- login as root
- /sbin/init 3 (to turn off the X-server)
- ./NVIDIA-Linux-x86_64-285.05.09.run
- reboot
- ./cuda.sh to give /dev/nvidia* correct permissions

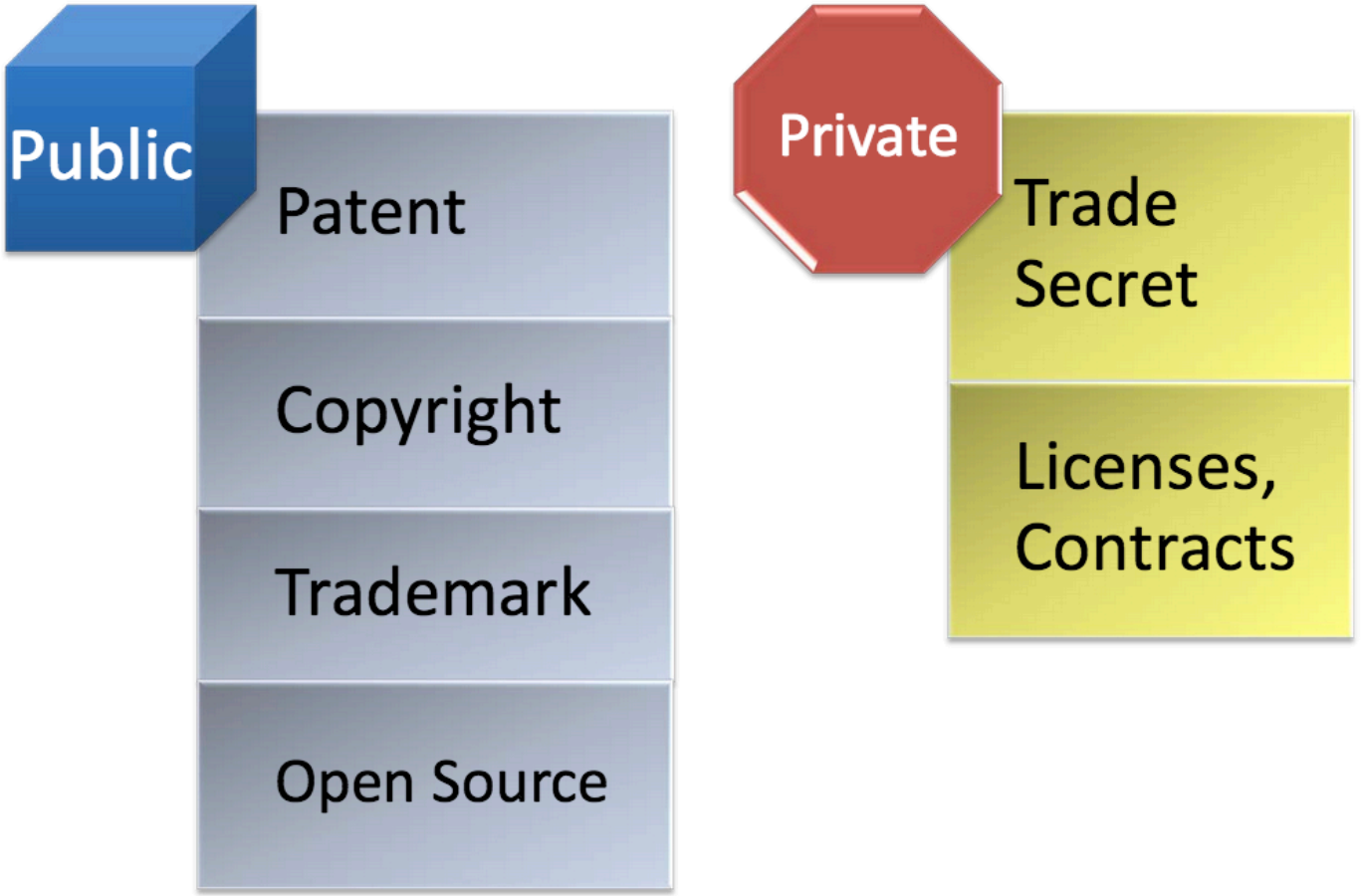
- tar –xzf fastrocs-1.3.1-RHEL5-x64-OpenCL-1.1-CUDA-4.1.tar.gz
- openeye/bin/ShapeDatabaseServer.py database.oeb.gz
- openeye/bin/ShapeDatabaseClient.py localhost:8080 query.sdf out.sdf

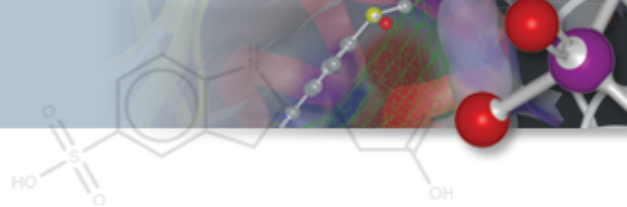
This is even worse!

fastrocs-1.3.1-RHEL5-x64-OpenCL-1.1-CUDA-4.1.tar.gz

NVidia OpenCL binaries are tightly locked to a particular driver version

Protecting our investment





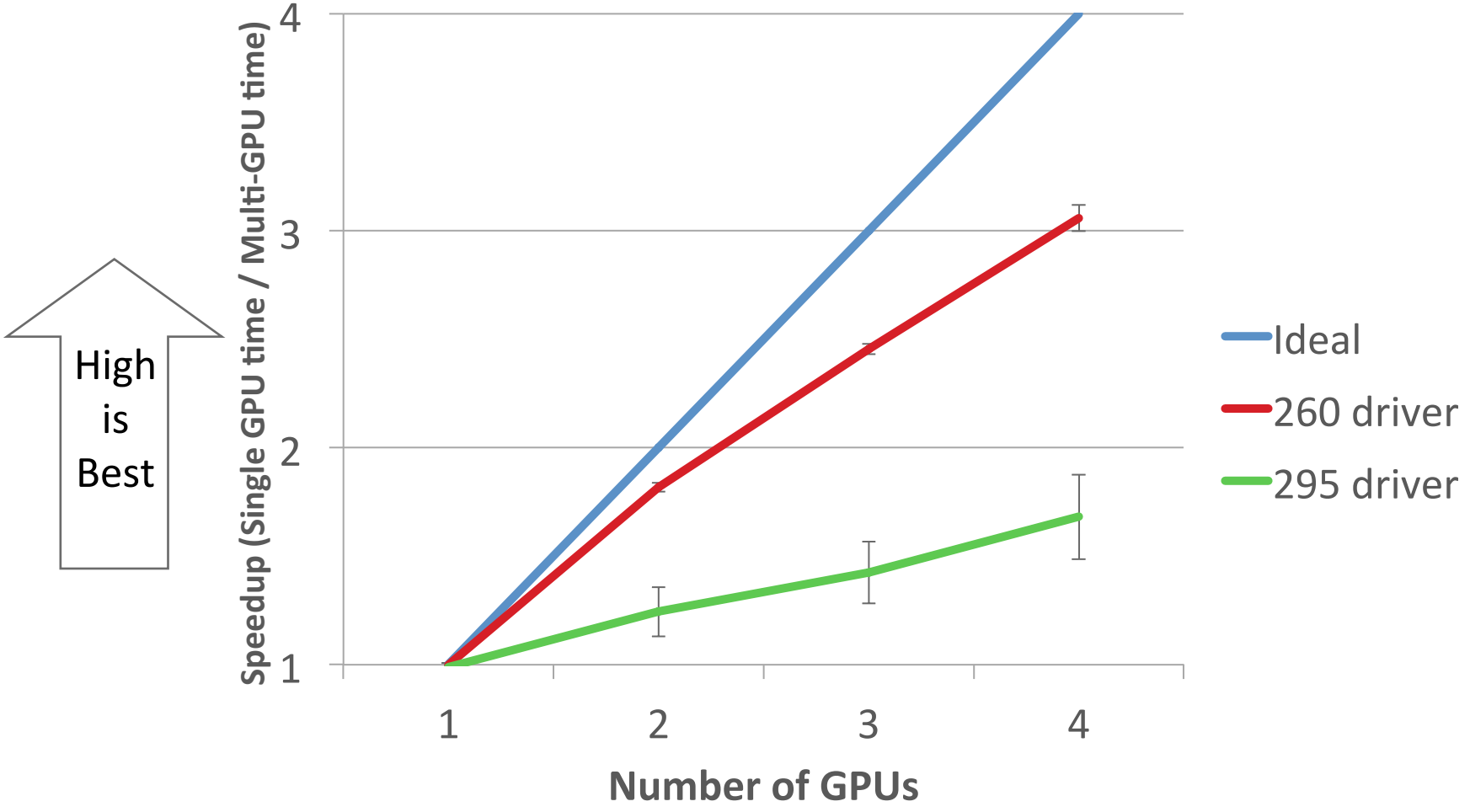
Trade secrets require due diligence

- We can't ship embedded OpenCL source code
- OpenCL SPIR is a great solution to this
  - Please support it!
- For now
  - Compile towards a specific NVidia driver
    - clCreateProgramWithBinary

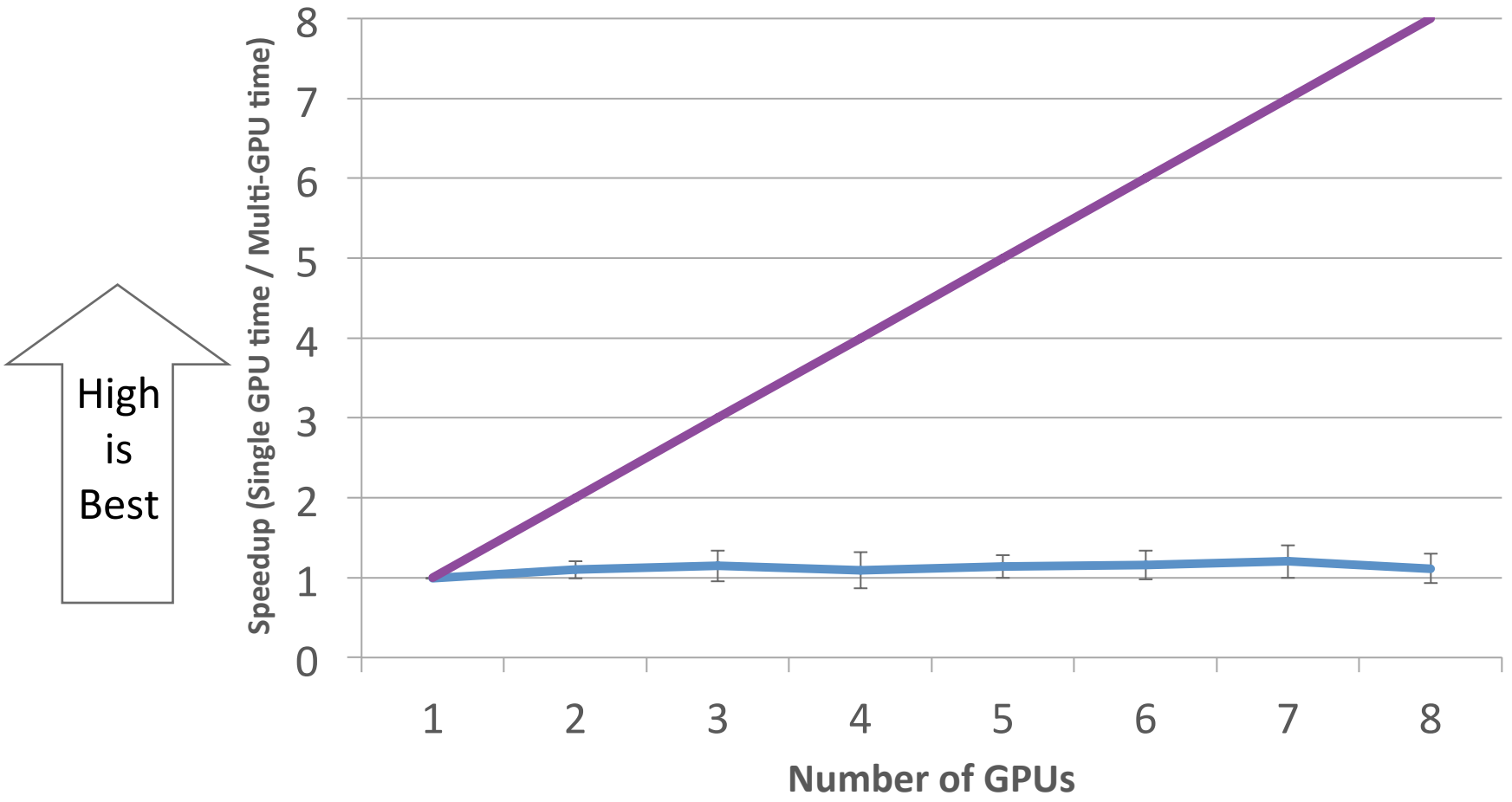
What's in the "chasm"?

- "GPUs are hard to deploy"
- "GPUs are hard to maintain"
- "The results aren't bitwise comparable"
- "There's nothing else to run on the GPU"

Scalability between drivers (4x C2050)



Really bad for 8x M2090



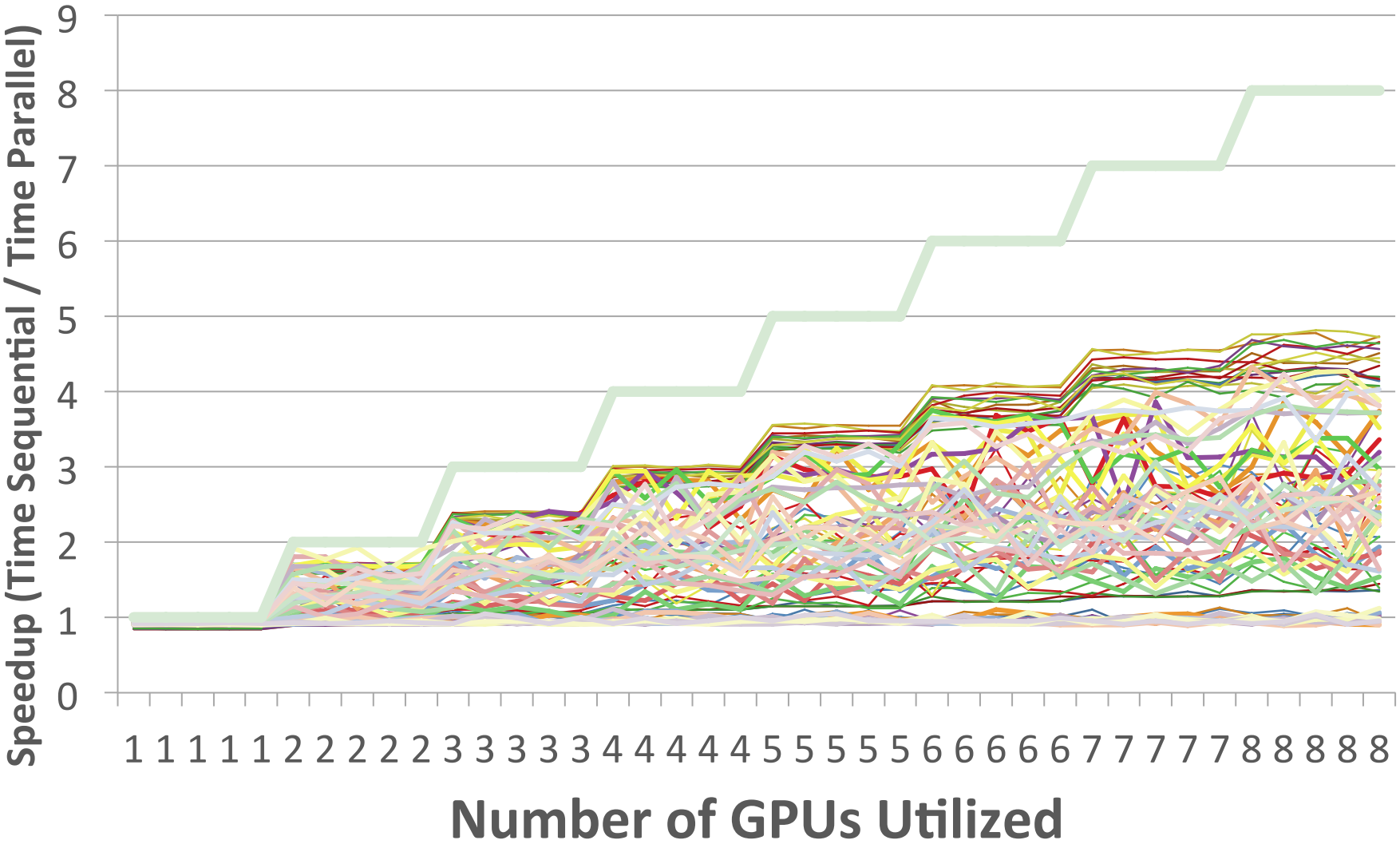
Ways to transfer to device

- CL_MEM_USE_HOST_PTR
  - kernelBuf = clCreateBuffer(CL_MEM_USE_HOST_PTR)
- CL_MEM_ALLOC_HOST_PTR|CL_MEM_COPY_HOST_PTR
  - kernelBuf = clCreateBuffer(CL_MEM_ALLOC_HOST_PTR|CL_MEM_COPY_HOST_PTR)
- CL_MEM_ALLOC_HOST_PTR
  - kernelBuf = clCreateBuffer(CL_MEM_ALLOC_HOST_PTR) - cacheable
  - ptr = clEnqueueMapBuffer(kernelBuf, CL_MAP_WRITE)
  - memcpy(ptr, data)
  - clEnqueueUnmapMemObject(ptr)
- clEnqueueMapBuffer
  - kernelBuf = clCreateBuffer() - cacheable
  - ptr = clEnqueueMapBuffer(kernelBuf, CL_MAP_WRITE)
  - memcpy(ptr, data)
  - clEnqueueUnmapMemObject(ptr)
- clEnqueueWriteBuffer
  - kernelBuf = clCreateBuffer() - cacheable
  - clEnqueueWriteBuffer(kernelBuf, data)
- oclCopyCompute
  - pinnedBuf = clCreateBuffer(CL_MEM_ALLOC_HOST_PTR|CL_MEM_READ_WRITE) – cacheable
  - pinnedPtr = clEnqueueMapBuffer(pinnedBuf, CL_MAP_WRITE) – cacheable
  - memcpy(pinnedPtr, data)
  - kernelBuf = clCreateBuffer() – cacheable
  - clEnqueueWriteBuffer(kernelBuf, pinnedPtr)

Ways to transfer from device

- CL_MEM_ALLOC_HOST_PTR
  - kernelBuf = clCreateBuffer(CL_MEM_ALLOC_HOST_PTR) - cacheable
  - ptr = clEnqueueMapBuffer(kernelBuf, CL_MAP_WRITE)
  - memcpy(data, ptr)
  - clEnqueueUnmapMemObject(ptr)
- clEnqueueMapBuffer
  - kernelBuf = clCreateBuffer() - cacheable
  - ptr = clEnqueueMapBuffer(kernelBuf, CL_MAP_WRITE)
  - memcpy(data, ptr)
  - clEnqueueUnmapMemObject(ptr)
- clEnqueueReadBuffer
  - kernelBuf = clCreateBuffer() - cacheable
  - clEnqueueWriteBuffer(kernelBuf, data)
- oclCopyCompute
  - pinnedBuf = clCreateBuffer(CL_MEM_ALLOC_HOST_PTR|CL_MEM_READ_WRITE) – cacheable
  - pinnedPtr = clEnqueueMapBuffer(pinnedBuf, CL_MAP_WRITE) – cacheable
  - memcpy(pinnedPtr, data)
  - kernelBuf = clCreateBuffer() – cacheable
  - clEnqueueReadBuffer(kernelBuf, pinnedPtr)

FastROCS scalability across 8x M2070



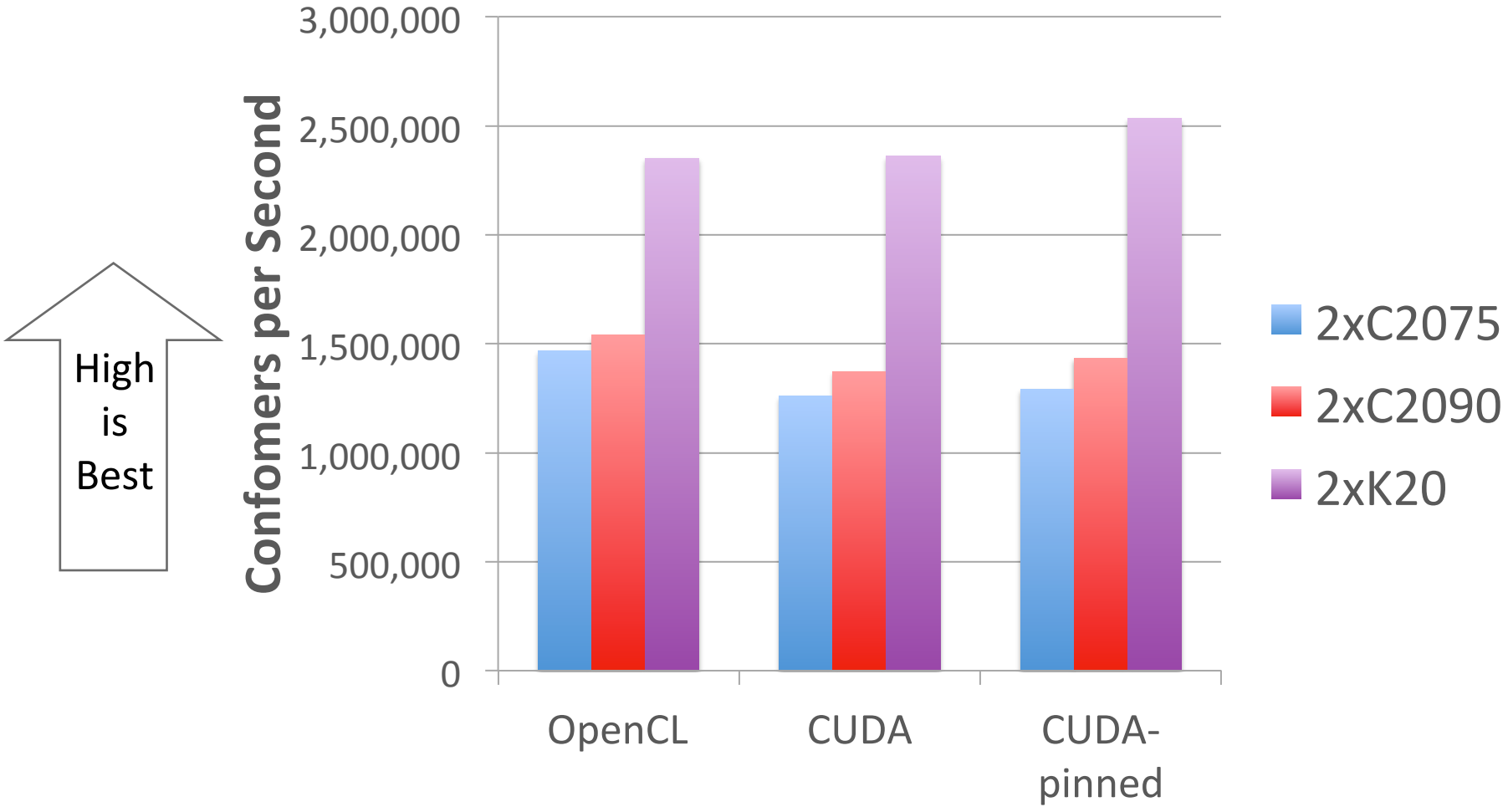
Lessons from the mess

- clEnqueueWriteBuffer > clEnqueueMapBuffer
- clEnqueueMapBuffer >> clEnqueueReadBuffer
- CL_MEM_* constants aren't worth the effort

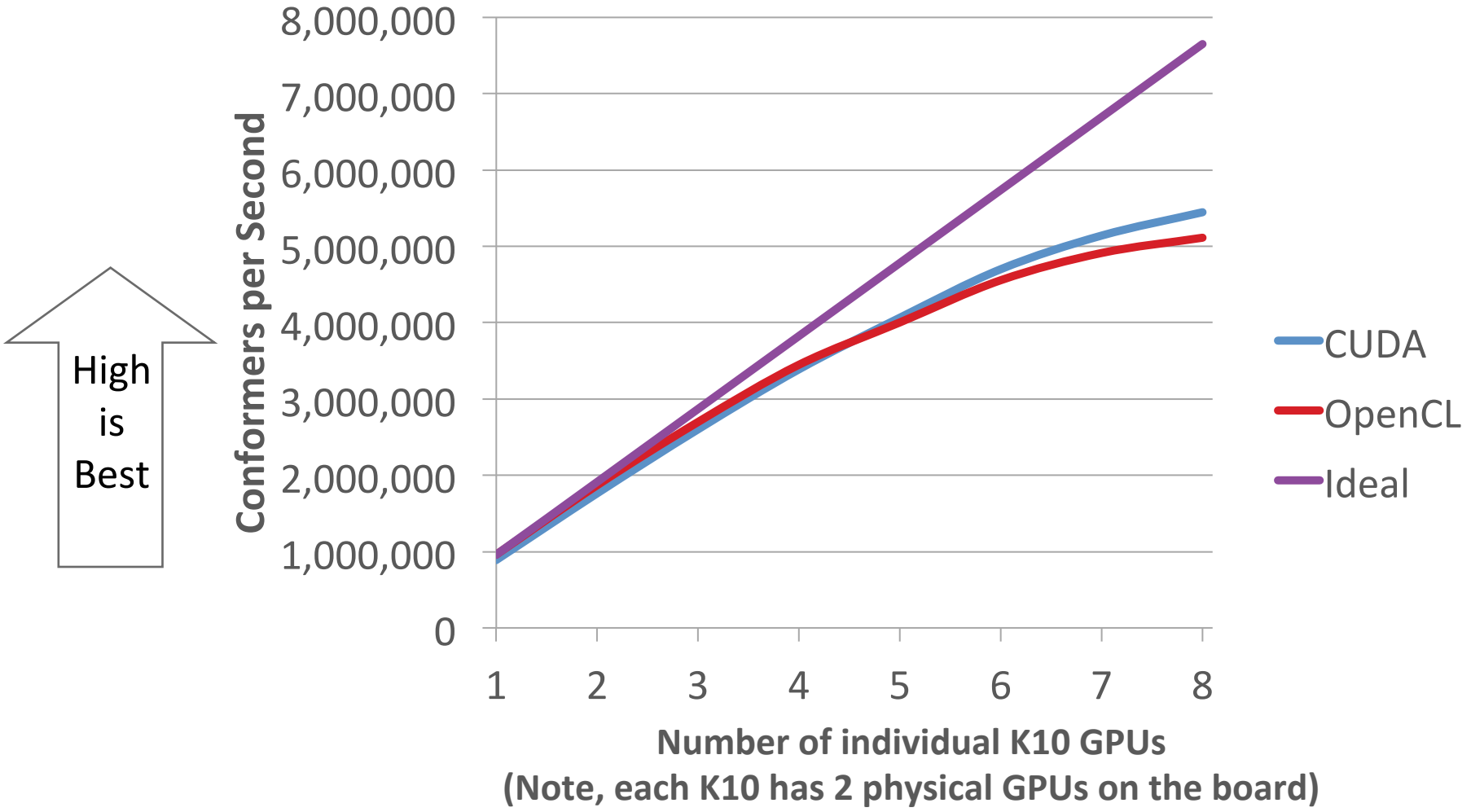
CUDA?

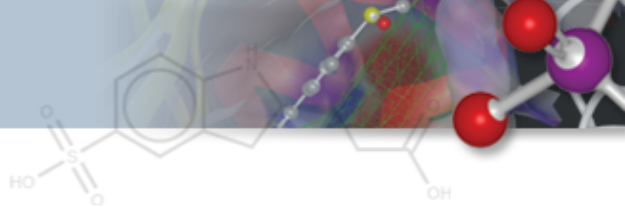
- Serious customers will only use NVidia cards
- Pinned memory
- Better support for binaries and compatibility
- CUDA support >> OpenCL support

FastROCS CUDA port



CUDA Scaling?



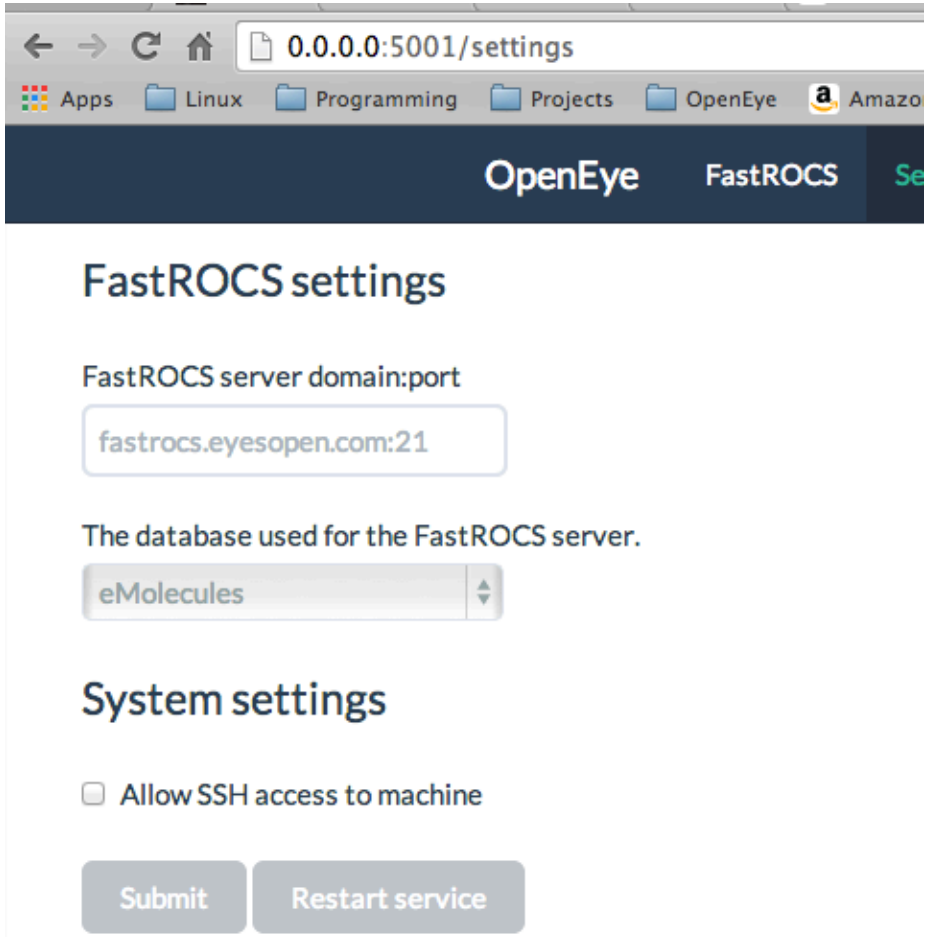


CUDA vs OpenCL: Ding Ding!

- Portability vs Innovation
- NVidia vs Intel and AMD
- Open vs Proprietary

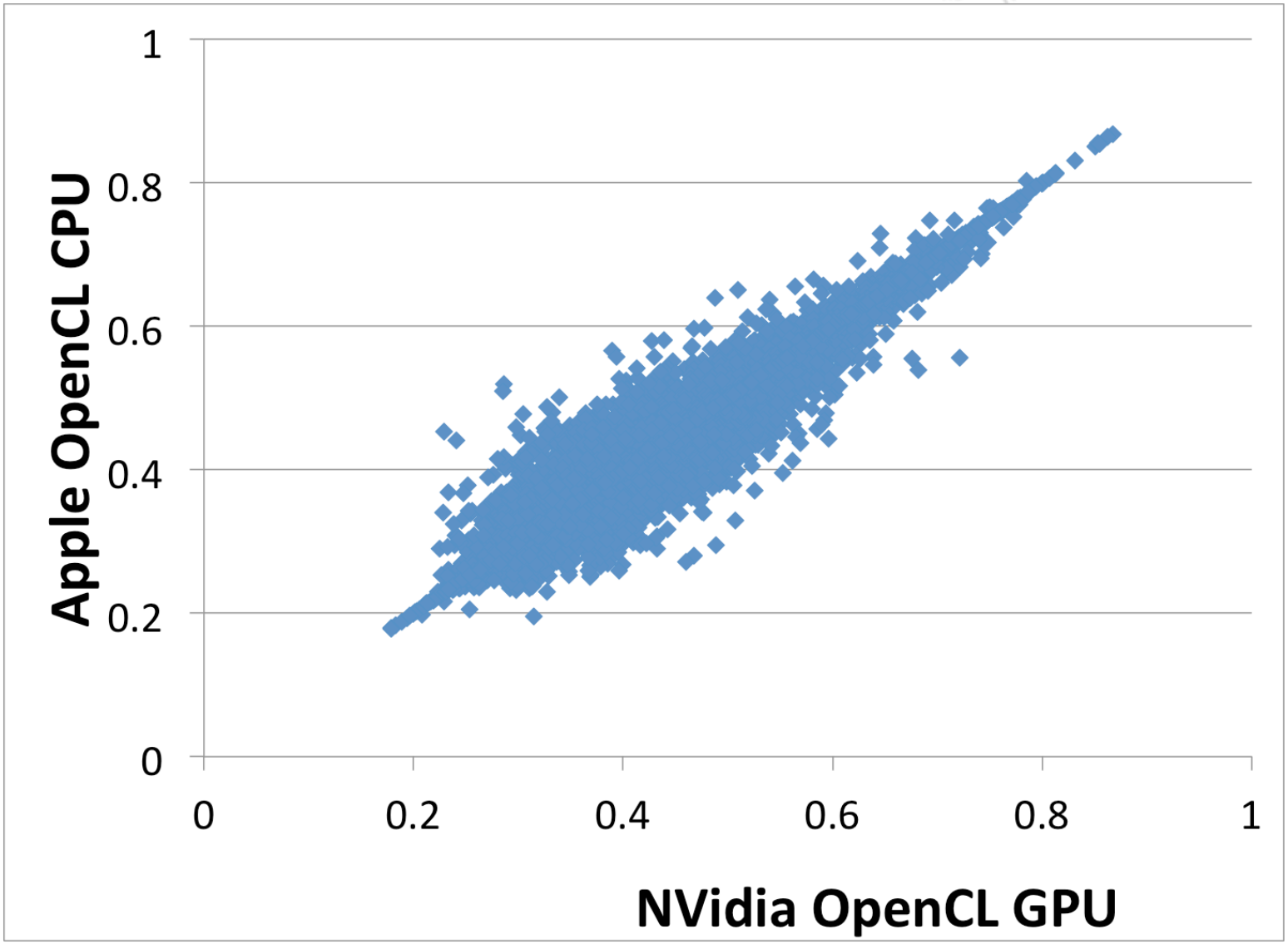
- Customers don't care…

FastROCS Instrument



What's in the "chasm"?

- "GPUs are hard to deploy"
- "GPUs are hard to maintain"
- "The results aren't bitwise comparable"
- "There's nothing else to run on the GPU"



Wish list

- Numerical analysis profiler
- Profile what is creating the most numerical error
- Understand the realized dynamic range
  - Should I use half, float, double, long double?

What's in the "chasm"?

- "GPUs are hard to deploy"
- "GPUs are hard to maintain"
- "The results aren't bitwise comparable"
- "There's nothing else to run on the GPU"

OpenEye product sheet

- AFITT
- BROOD
- FRED
- EON
- OMEGA
- pKa Prospector
- QUACPAC
- ROCS
- SZMAP
- SZYBKI
- VIDA

- OEChemTK
- DepictTK
- DockingTK
- GraphemeTK
- GraphSimTK
- LexichemTK
- MedChemTK
- MolPropTK
- OmegaTK
- QuacPacTK
- SpicoliTK
- SzmapTK
- SzybkiTK
- ZapTK

C++

python™

Java™

Microsoft C#.net

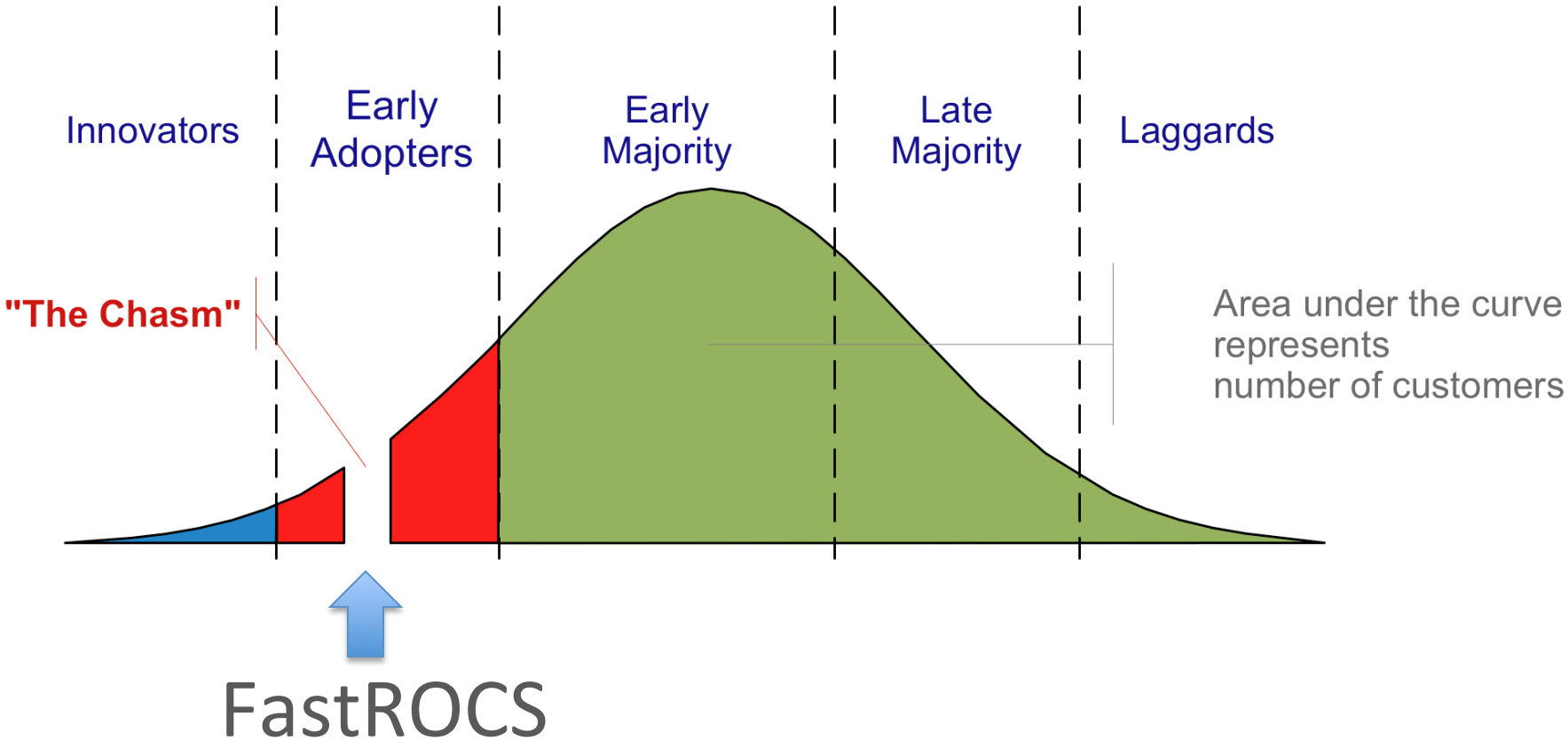
C++ future

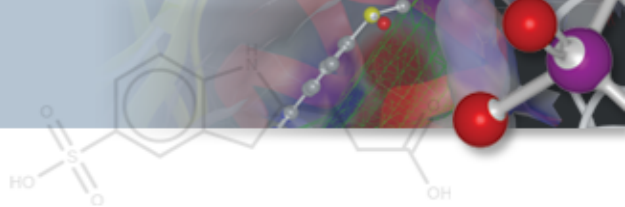
- SYCL looks great
  - C++ -> OpenCL SPIR is promising
- C++ working group on parallelism is promising
  - But 10 years till usable
- Need to make accelerator computing ubiquitous
  - Convert to the church of Eric Berdahl

What's in the "chasm"?

- "GPUs are hard to deploy"
  - Mature-ish
- "GPUs are hard to maintain"
  - Maturing
- "The results aren't bitwise comparable"
  - Good challenge for academia
- "There's nothing else to run on the GPU"
  - Good challenge for everyone here

Technology Adoption Lifecycle

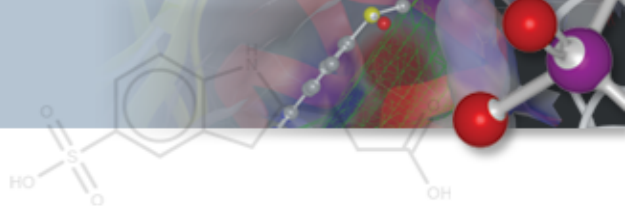




OpenEye Scientific Software

For more information, please contact us:

business@eyesopen.com
support@eyesopen.com
www.eyesopen.com
+1-505-473-7385



Acknowledgements

- Dave Mullally (HP)
  - For access to test hardware
- Nikolai Sakharnykh (NVidia)
  - For a great effort porting to CUDA
- Herb Sutter (Microsoft)
  - For insight about Little's Law