smarter chemistry | smarter decisions™

Porting a commercial application to OpenCL: a case study

Simon Krige, Mark Mackey

Simon McIntosh-Smith, Richard Sessions

IWOCL, May 2013

Agenda

- Background
- About **blaze**V10
- Porting commercial code to OpenCL
- Kernel code Optimizations
- Host code Optimizations
- **blaze**V10 GPU benchmark
- Science Advantages of GPUs
- Conclusions

About me

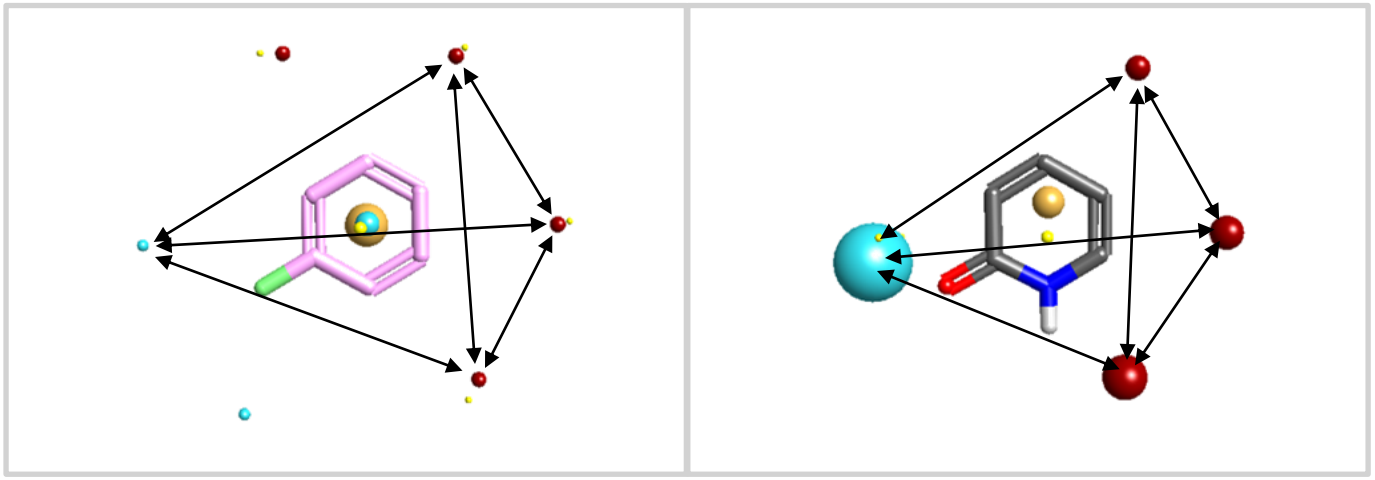
- Originally from Geneva, Switzerland.
- Graduated with a Masters degree in computer science from the University Of Bristol two years ago.

- Working on an 18 months project at Cresset in collaboration with the University of Bristol, funded by the Knowledge Transfer Partnership
- About 14 months into the project now!

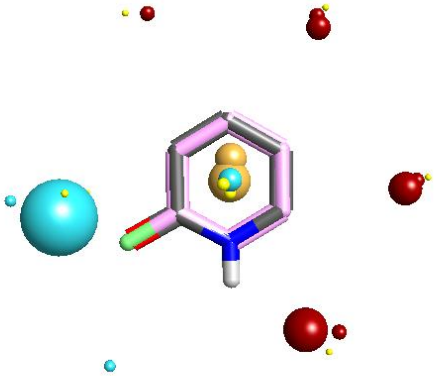
About Cresset

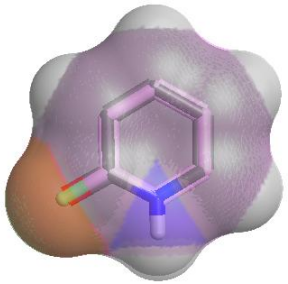
- Founded in 2002 by Dr Andy Vinter
- Use shape and electrostatics of ligands to compare molecules in 3D
- Software
  - Ligand based virtual screening
  - Develop pharmacophores and understand structure activity relationships
  - Find novel bioisosteric replacements for parts of your molecule
- Services
  - Full range of computational chemistry services

Non-Classical Molecular Comparisons



Fields
0.66

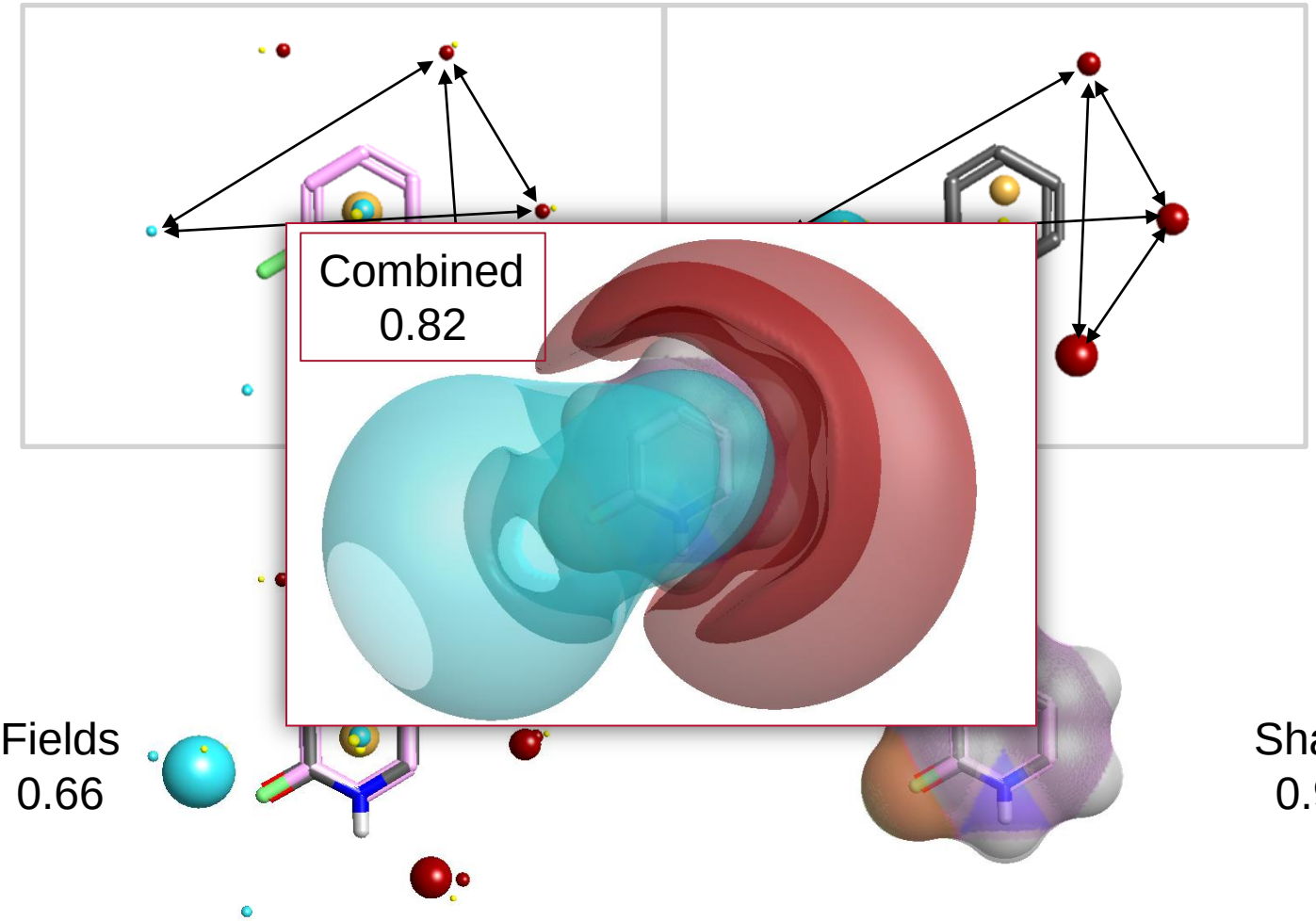




Shape
0.98

Cheeseright et al, *J. Chem Inf. Mod.*, 2006, 665

Non-Classical Molecular Comparisons

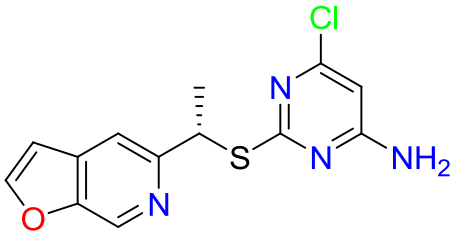


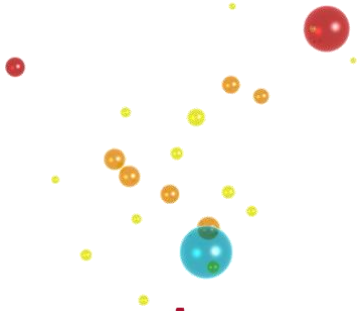
Cheeseright et al, *J. Chem Inf. Mod.*, 2006, 665

blazeV10

- Ligand based virtual screening - search a database with a query structure, retrieve a hit list
- Runs on a Linux cluster
- Can screen ~5 million compounds in a few hours
  - 100→500 cpu cluster
  - i.e. a high number of CPUs working together
- We would like a cheaper and faster solution!

blazeV10 compound search





Search:
Quick
Thorough
Exhaustive

Database of compound conformations

Title	score
ZINC03833947	0.8975
ZINC03591572	0.8289
ZINC03833950	0.8250
ZINC03591562	0.8099
ZINC03833921	0.8042
ZINC03833899	0.7987
ZINC03591551	0.7965
ZINC03833941	0.7889
ZINC03591586	0.7857
ZINC03833922	0.7857
ZINC03833948	0.7837
ZINC03833927	0.7817

We want to do this as fast as possible!

Porting commercial code to OpenCL

- **blaze**V10 consists of approximately 120,000 lines of FORTRAN77
- After profiling the serial code we found 3 algorithms that need porting to OpenCL:
  - Clique matching algorithm: ~500 lines of code
  - Field point similarity algorithm: ~300 lines of code
  - Gaussian volume overlap algorithm: ~300 lines of code
- Each OpenCL kernel is ~400 lines of code – fairly large kernels!
- Why OpenCL? At Cresset we value portability of our software highly
  - Our clients have different hardware resources, we want to make sure we can provide all of them with a GPU accelerated solution

OpenCL Optimizations

- Kernel code optimizations:
  - Memory coalescence
  - Local and private memory
  - Branch divergence
- Host code optimizations:
  - Work group size
  - Overlapping Compute and I/O
  - Kernel code integration and obfuscation
  - Drivers and tools

Memory Coalescence

- Each work group executes kernel code in a SIMD fashion
- Memory accesses within a work group a grouped together
  - Want to minimize the batched requests to global memory as much as possible
- We need work items to be accessing memory addresses that are spatially close to each other
- Structures of arrays vs. arrays of structures
- Requires you to think differently from when writing serial code!

Local and private memory

cresset

- Making effective use of the GPU's memory hierarchies is vital in achieving good performance
- This has been one of the most critical issues in making our code performance portable
- Currently not using any local memory, only global and private

Branch divergence

- Branches can have a large impact on GPU utilisation
- Try to avoid them as much as possible, especially in loops
- If a data-dependant branch is encountered, each branch path has to be executed serially by the work group
- Predication is used to “switch instructions off”

(1) If statement

```
int id = get_global_id(0);
if (a[ id ] > 0) {
    pos ++;
}
```

(2) If-else statement

```
int id = get_global_id(0);
if (a[ id ] > 0) {
    pos ++;
} else {
    pos --;
}
```

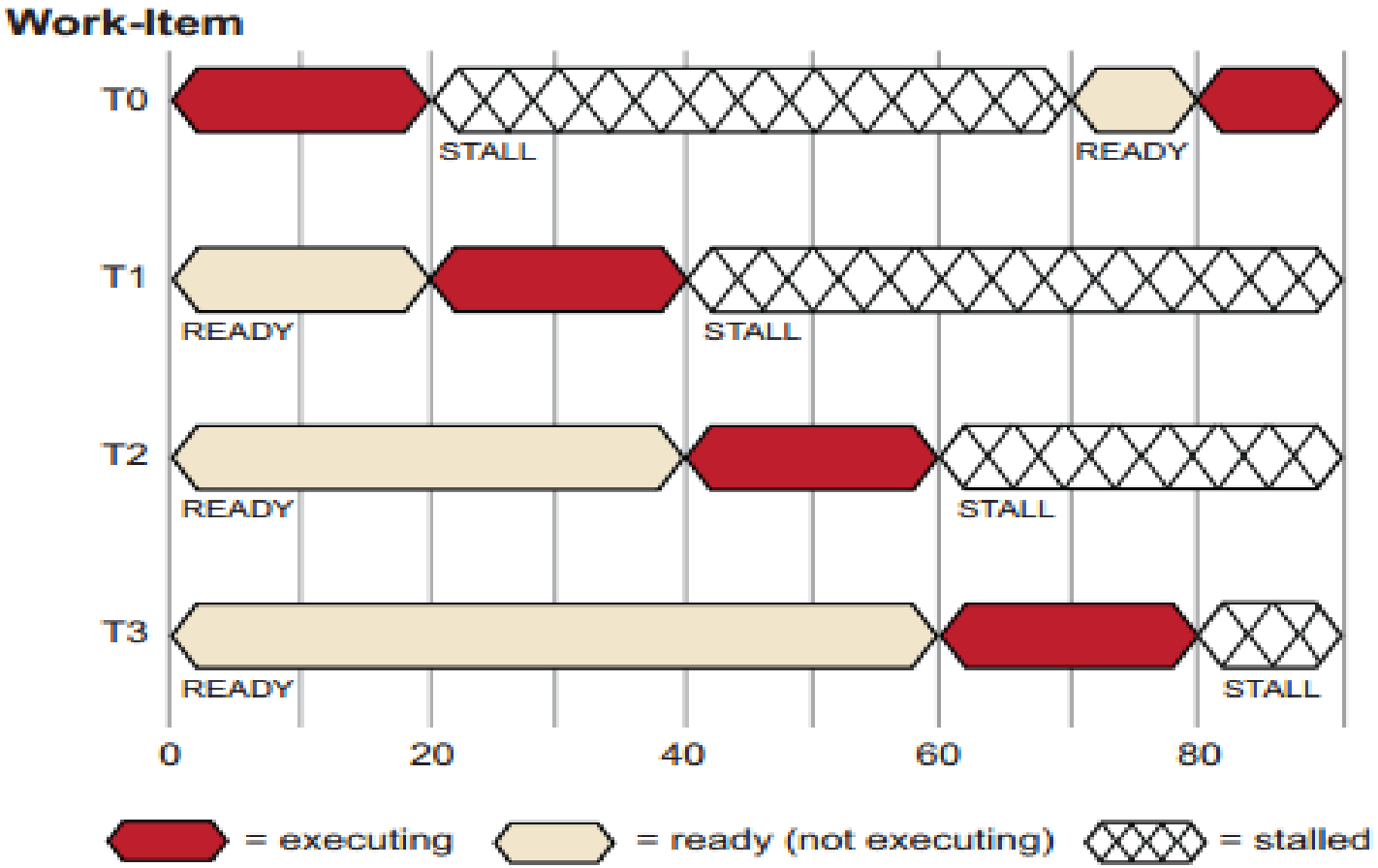
(3) Loop with variable trip-count

```
int pos = a[get_global_id(0)];
for(int i=0; i < pos; pos++) {
    // do work
}
```

Work group sizes

- Preferred work group size should be the smallest work group size you set
- Keep a low multiple for ALU bound kernels
- Use a multiple of 4 or 8 for memory bound kernels
  - This will hide memory bottlenecks: other kernels in work group can execute while one stalls on a memory access
- Better solution is to run a small set of benchmarks/auto-tuning
  - Ideal work group size can vary on the same device, depending on the problem

Execution of work-items on a single PE



* AMD Accelerated Parallel Processing
OpenCL

Overlapping Compute and I/O

- We always want to have all data readily available for the next GPU computation to start
- I/O can sometimes be a bottleneck
- The solution is to have multiple threads on the host
- One thread deals with I/O
- For each OpenCL device, a threads gets data for computation, creates buffers, and executes kernels
- This method also allows us to control how much host memory is used at any one time
- Stop parsing data when there is enough data in memory to keep devices busy!

Kernel code integration and obfuscation

- Writing closed source OpenCL code is not trivial
- Usually code is stored in external *.cl* files
- Solution (1)
  - Parse code in .cl files and generate encrypted string
  - Store encrypted string in a header file
  - Decrypt before you pass it to kernel compile function
  - Problem: a clever hacker could set a breakpoint at kernel compile function and read source string
- Solution (2)
  - Compile all kernels to object files and load them at runtime
  - Problem: need to compile for all platforms – less flexibility
  - Problem: need to transfer object files along with executable

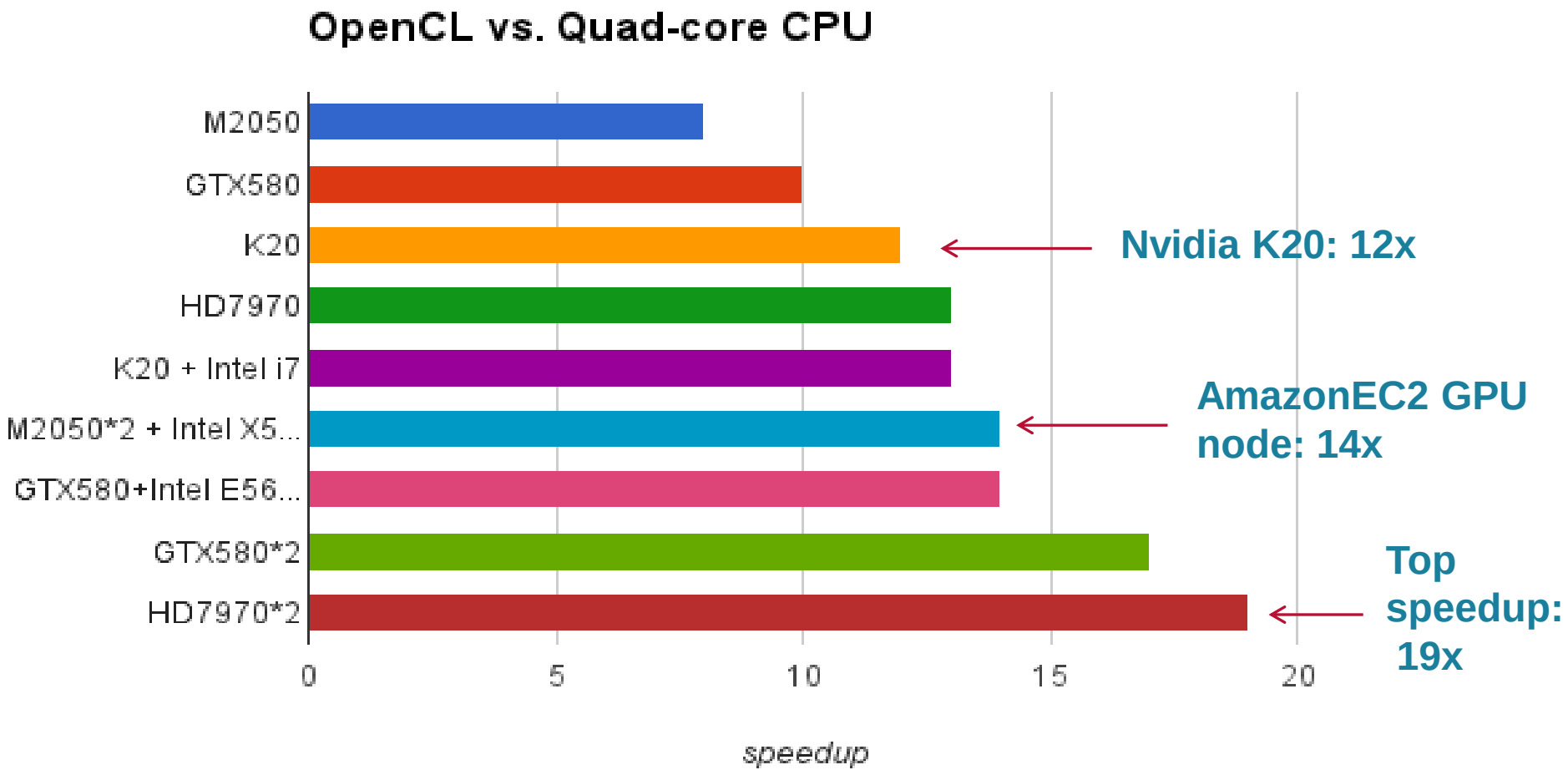
Drivers and tools

- It is important to update drivers, OpenCL SDK and tools as often as possible
- Never assume that OpenCL host code and kernel compilation will behave in the same way with different vendors
- In a commercial environment we want to test our OpenCL code on as many platforms and drivers as possible

blazeV10 GPU benchmark

- Screened one ligand against 2500 molecules – 300k conformations
- Standard instance: 48 conformations processed per second on a quad-core Intel® Core i7-3770 CPU @ 3.40GHz
- CPUs:
  - Intel® Core i7-3770 CPU @ 3.40GHz (4 cores - 4 threads)
  - Intel® Xeon CPU X5570 @ 2.93GHz (4 cores – 8 threads)
  - Intel® Xeon CPU E5645 @ 2.40GHz (6 cores - 12 threads)
- GPGPUs:
  - NVIDIA GTX580
  - AMD HD7970
- HPC GPUs:
  - NVIDIA M2050
  - NVIDIA K20
- CPUs and GPUs will work together, it's a heterogeneous world!

blazeV10 GPU benchmark



Science Advantages

- Faster Virtual screening
  - Easier deployment
  - Cheaper
  - Desktop box with 4 GPUs vs. 150 core cluster
- New science
  - Using multiple molecule 3D comparisons in new ways
  - Similarity matrices
- Easier
  - Manage fewer instances on AmazonEC2
- Accurate
  - Results accuracy is preserved i.e. we are not sacrificing accuracy for speed.

Conclusions

- GPU computing has an important role to play in the pharmaceutical/science industry
  - They will probably become an essential tool in the future of sustainable scientific research
- GPUs and OpenCL are mature enough for use in a commercial environment
  - But still some way to go!

Acknowledgements

- Bristol
  - Prof. Simon McIntosh-Smith
  - Prof. Richard Sessions
- Cresset
  - Dr. Mark Mackey
  - Dr. Tim Cheeseright
- Nvidia
  - Mark Berger for donation of a K20 GPU
- AMD
  - Lee Howes for helping fix problems on HD7970

Thank you

simon@cresset-group.com


Cresset Group


Cressetgroup


Cresset

www.cresset-group.com