# GPUs for Scientific Computing

Mike Giles

mike.giles@maths.ox.ac.uk

Oxford-Man Institute of Quantitative Finance

Oxford University Mathematical Institute

Oxford e-Research Centre

University of Bristol HPC Symposium, Oct 21, 2009

# Computing – Recent Past

- driven by the cost benefits of massive economies of scale, specialised chips (e.g. CRAY vector chips) died out, leaving Intel/AMD dominant
- Intel/AMD chips designed for office/domestic use, not for high performance computing
- increased speed through higher clock frequencies, and complex parallelism within each CPU
- PC clusters provided the high-end compute power, initially in universities and then in industry

- at same time, NVIDIA and ATI grew big on graphics chip sales driven by computer games

# Computing – Present/Future

- move to faster clock frequencies stopped due to high power consumption
- big push now is to multicore at (slightly) reduced clock frequencies

- graphics chips have even more cores (up to 240) – big new development here is a more general purpose programming environment

  Why? Initially, because computer games do increasing amounts of “physics” simulation on the GPU, but branching out into other areas.

# CPUs vs GPUs

- CPUs have up to 6 cores on a single chip, and 10-20 GB/s bandwidth to main system memory
- NVIDIA GPUs have up to 240 cores on a single chip, and 100+ GB/s bandwidth to graphics memory
- offer 50–100$\times$ speedup relative to a single CPU core
- roughly 10$\times$ speedup relative to two quad-core Xeons (if not using SSE instructions)
- also 10$\times$ improvement in price/performance and energy efficiency

How is this possible? Logically simpler cores (SIMD, no out-of-order execution, minimal caching) suitable for vector computing, not for general purpose computing

# GPUs

Is this sustainable? Yes!

- IBM, AMD and Intel all producing GPUs
- NVIDIA has good headstart on software side with CUDA environment
- new OpenCL software standard (based on CUDA and pushed by Apple) will probably run on all platforms
- driving applications are:
  - computer games “physics”
  - video (e.g. HD video decoding)
  - computational science
  - computational finance
  - oil and gas

# GPUs

- NVIDIA:
  - GTX graphics cards for “consumers” – £100-400
  - C1060 for scientific computing – no graphics output but 4GB of graphics memory and costs about £800
  - next-generation chip out soon – 512 SP cores
- IBM: Cell processor is expensive and hard to program – future development may have been cancelled
- AMD: has good hardware, but poor software – will support the new OpenCL standard
- Intel: developing its own GPU code-named “Larrabee” – 2nd-gen in 2011 will have compute capability

# NVIDIA GPUs

- basic building block is a “multiprocessor” with 8 cores, 16384 registers and a small amount of shared memory
- different chips have different numbers of these:

| product | multiprocs | bandwidth | cost |
|---|---|---|---|
| GTX 260 | 27 | 112GB/s | £140 |
| GTX 295 | $2 \times 30$ | $2 \times$112GB/s | £360 |

- each card also has 1-4GB graphics memory for:
  - global memory accessible by all multiprocessors
  - special read-only constant memory
  - additional local memory for each multiprocessor

# NVIDIA GPUs

The 8 cores in a multiprocessor are SIMD (Single Instruction Multiple Data) cores:

- all cores execute the same instructions simultaneously
- vector style of programming harks back to CRAY vector supercomputing
- natural for graphics processing and much scientific computing
- SIMD is also a natural choice for massively multicore to simplify each core
- requires specialised programming (no standard)

# CUDA programming

CUDA is NVIDIA's program development environment:

- based on C with some extensions
- lots of example code and good documentation
  – 2-4 week learning curve for those with experience of OpenMP and MPI programming
- growing user community active on NVIDIA forum
- main process runs on host system (Intel/AMD CPU) and launches multiple copies of "kernel" process on graphics card
- communication is through data transfers to/from graphics memory
- minimum of 4 threads per core, but more is better

# CUDA programming

How hard is it to program?

Needs combination of skills:

- splitting the application between the multiple multiprocessors is similar to MPI programming, but no need to split data – it all resides in main graphics memory
- SIMD CUDA programming within each multiprocessor is a bit like OpenMP programming – needs a good understanding of memory operation (but that's becoming less critical over time)
- difficulty also depends a lot on application

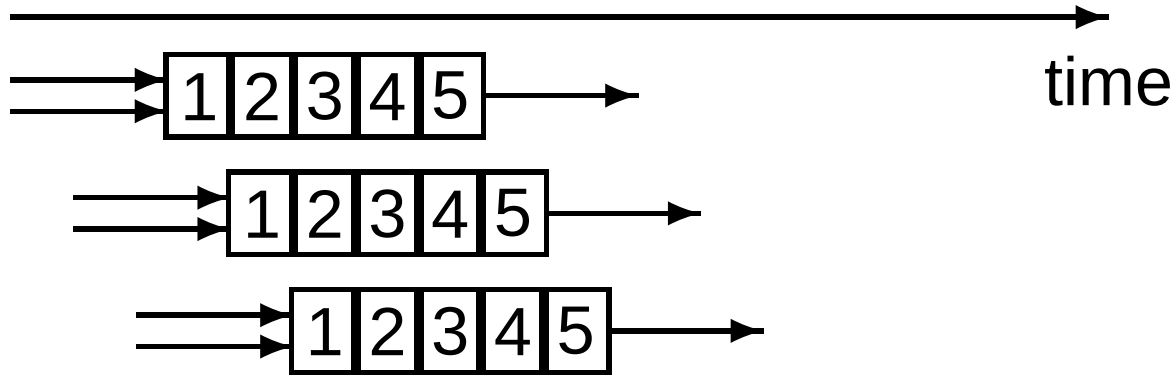
# NVIDIA multithreading

Lots of active threads is the key to high performance:

- no “context switching”; each thread has its own registers, which limits the number of active threads
- threads execute in “warps” of 32 threads per multiprocessor (4 per core) – execution alternates between “active” warps, with warps becoming temporarily “inactive” when waiting for data

# NVIDIA multithreading

- for each thread, one operation completes long before the next starts – avoids the complexity of pipeline overlaps which can limit the performance of modern processors




- memory access from device memory has a delay of 400-600 cycles; with 40 threads this is equivalent to 10-15 operations and can be managed by the compiler

# My experience

- Random number generation (mrg32k3a/Normal):
  - 2500M values/sec on GTX 280
  - 70M values/sec/core on Xeon using Intel's VSL
- LIBOR Monte Carlo testcase:
  - 180x speedup on GTX 280 compared to single thread on Xeon
- 3D PDE application:
  - factor 50x speedup on GTX 280 compared to single thread on Xeon
  - factor 10x speedup compared to two quad-core Xeons

GPU results are all single precision – double precision is currently 2-4 times slower, no more than factor 2 in future

# NAG Numerical Routines for GPUs

Random number generation:

- Pseudo-random (L'Ecuyer's `mrg32k3a`) and quasi-random (`sobol`) generation
- uniform, exponential, Normal and gamma output distributions
- available free to academics under a development license
- being assessed by a number of banks
- lots of interest at conferences and industry events

Other capabilities will be added in the future.

# What is needed now?

Skilled manpower, training:

- 50+ on Oxford CUDA mailing list: students and post-docs in almost all science departments
- 1-week CUDA course this summer
- in 3 years time, many PhDs in computational science will have these skills

More development of libraries, high-level packages:

- Monte Carlo simulation
- PDE solvers
- …

# Further information

LIBOR and finite difference test codes
`www.maths.ox.ac.uk/~gilesm/hpc/`

Lecture notes and practicals
`www.maths.ox.ac.uk/~gilesm/cuda/`

NAG Numerical Routines for GPUs
`www.nag.co.uk/numeric/GPUs/`

NVIDIA's CUDA homepage
`www.nvidia.com/object/cuda_home.html`

**Announcement: 1st UK CUDA Academic Developers Meeting, Dec 7th in Oxford**