

GPUs and Emerging Computer Architectures




Mike Giles

Oxford e-Research Centre
Mathematical Institute

Oxford-Stanford Conference on Big Data

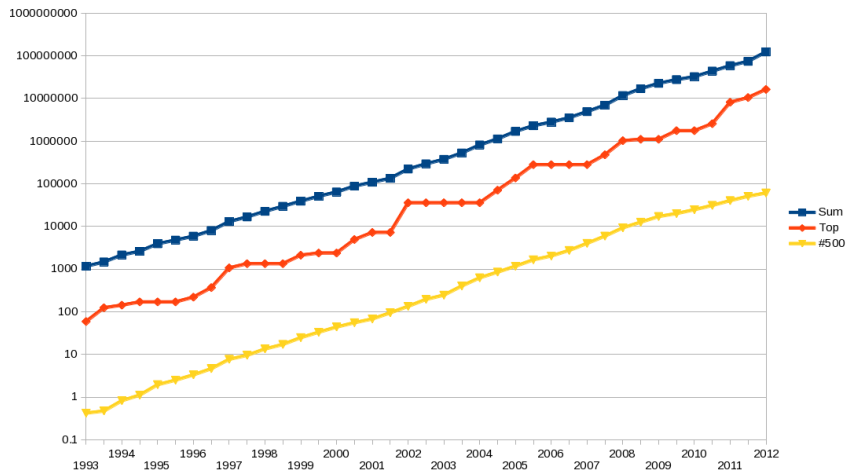
Nov 28–29, 2012

Outline

- more power 
- more complexity 
- more help 

Good news

Computing power is continuing to increase – Top500 data



Really impressive – 300× more capability in 10 years!

Bad news

Writing software to exploit that capability is getting much more complex

1) massive parallelism

- circuitry keeps getting smaller \implies much more circuitry on each chip
- energy efficiency is now very important \implies
can't get more performance by higher clock frequency
- hence, performance now comes through massive parallelism
(2688 cores on latest GPUs)
- clock frequency has even come down a little to reduce energy consumption per flop

Bad news

2) vector processing

- don't want to dedicate a large fraction of the chip to "command & control"
- instead, cores work in small groups, all doing the same instruction at the same time, but on different data
- similar to old days of vector computing on CRAY supercomputers, but with fixed length small vectors (length 32 on NVIDIA GPUs)
- not just on GPUs – CPUs also have vectors units (SSE, AVX) which are getting longer (4 / 8 on most, but 8 / 16 on Intel's Xeon Phi)
- this is tricky for algorithms with lots of conditional branching, but there are various algorithmic tricks that can be used

Bad news

3) data movement is often key

- takes much more energy and much more time to move data, even from one side of a chip to the other, than to perform a floating point operation
- increasingly, the cost (energy, time) of an algorithm depends on how much data is moved, not the number of floating point operations
- in some cases, this needs a fundamental re-think about the algorithms we use for different applications, or at least a re-think about the way in which we implement them

Bad news

4) we're in a period of rapid hardware innovation . . .

- NVIDIA GPUs – latest have 2688 cores organised as 14 groups of 192 cores, with a vector length of 32
- mainstream Intel CPUs have up to 8 cores, each with an AVX vector unit of length 4/8
- Intel Xeon Phi is GPU-like, with up to 60 cores, each with an AVX vector unit of length 8/16
- AMD, IBM and others also have their own innovations

Bad news

5) ... and accompanying software diversity

- CUDA for NVIDIA GPUs
- OpenCL for NVIDIA/AMD GPUs, and also CPU vector units (but still limited use / support?)
- OpenACC for simpler high-level parallelism (but limited applicability?)
- variety of other approaches from Intel for exploiting CPUs:
 - ▶ low-level vector primitives
 - ▶ Cilk
 - ▶ SPMD

Help available

Back to good news – lots of help available to tackle this complexity

- many-core parallelism one of the big challenges in computer science
- scientific computing groups working on achieving good performance, and making it more accessible to application developers
- inter-disciplinary research institutes are linking everyone together
 - application developers no longer have to struggle on their own

Oxford – Oxford e-Research Centre

Stanford – Institute for Computational and Mathematical Engineering

– both are NVIDIA CUDA Centers of Excellence

Help available

Short-term challenge:

- experiment with different emerging architectures
- develop implementation techniques to extract the most performance
- determine which architectures are best for different applications
- develop library software to simplify application development

Help available

Longer-term challenge:

- application developers don't want to constantly re-write their codes for the latest hardware and software
- solution is for application developers to specify **what** they want done at a higher level, leaving it to parallel computing experts to determine **how** it should be done
- this is leading to development of high-level frameworks and domain-specific languages
- difficulty is in defining a high-level framework which addresses a sufficiently large and important class of applications, and getting funding to sustain development
- if successful, the work of a few computing experts supports a large number of applications

Help available

Some associated research topics:

- automated code generation and run-time compilation:
- auto-tuning, to optimise parameterised code, either on a test problem or through run-time optimisation
- tiling, a technique to reuse data more often within cache, and reduce the number of times it is brought in from main memory – a good example of potential benefits of high-level automated techniques

Help available

Current collaborations in Oxford:

- real-time processing of transients in radio-astronomy
- high-level framework for parallelisation of unstructured grid methods in computational engineering
- high-level framework for stochastic biochemical reaction simulations
- high-level framework for hidden Markov models in bioinformatics

We also have excellent facilities through Oxford Supercomputing Centre:

- Emerald: 372-GPU HP/NVIDIA system
- Iridis: 12,000-core IBM/Intel system

Big Data

The other side of HPC is storing / managing / processing huge amounts of data – this typically involves a different set of experts

There is a rapidly emerging Big Data community in Oxford, driven by applications in bioinformatics, genomics and structural biology – led by Robert Esnouf (Division of Structural Biology, and chair of OSC Technical Advisory Group)

Personally, I know it is very important, and I know how much I don't know (always a good starting point!)

With that caveat, I'll offer a few comments

Big Data

Hardware:

- similar progress to computing side
- parallelism is often important – a single disk drive is slow, but if you “stripe” the data across many of them the data-rate is much higher
- large-scale flash memory storage is becoming more popular – more expensive than disks but faster

Software:

- this is the area where I’m really ignorant
- huge amounts of “raw” data held in standard binary forms
- smaller amount of processed data / meta-data which often needs to be held in databases for querying – can then link back to the associated raw data
- issues of archiving, provenance and security also important

Conclusions

- computing power continues to increase ...
- ... but the complexity is increasing too
- application developers need help from scientific computing experts to
 - ▶ exploit the potential of new architectures
 - ▶ do so without huge programming efforts
- the good news is that we have the variety of experts and inter-disciplinary structures to work collaboratively on these problems
- important also to involve mathematicians – progress in numerical methods has delivered almost as much as progress in computers