An Introduction to OpenACC

Alistair Hart
Cray Exascale Research Initiative Europe

Timetable

- **Day 1: Wednesday 29th August 2012**
  - 13:00 Welcome and overview
  - 13:15 Session 1: An Introduction to OpenACC
    - 13:15 Lecture: The OpenACC programming model
    - 14:15 Practical: compiling and running a sample OpenACC code
    - 14:45 break
  - 15:15 Session 2: Accelerating a simple code
    - 15:15 Worked example: OpenACC-ing a simple code
    - 15:45 Practical: accelerating the simple code
  - 16:30 close

- **Day 2: Thursday 30th August 2012**
  - 09:00 Session 3: Accelerating a larger code
    - 09:00 Lecture: Preparing to OpenACC a code
    - 09:45 Worked example: OpenACC-ing a larger code
    - 10:15 Practical: preparing and accelerating a larger application
    - 10:45 break
  - 11:15 Presentation by Nvidia (Timothy Lanfear)
  - 12:30 lunch
  - 13:30 Session 4: Improving OpenACC performance
    - 13:30 Lecture: OpenACC performance tuning
    - 14:15 Practical: continuing to accelerate a larger code
    - 15:00 break
  - 15:30 Session 6: OpenACC for parallel applications
    - 15:30 Case study
  - 16:15 Summary and outlook
  - 16:30 close
Contents

● The aims of this course:
  ● To motivate why directive-based programming of GPUs is useful
  ● To introduce you to the OpenACC programming model
  ● To give you some experience in using OpenACC directives
    ● with some hints and tips along the way
  ● To introduce you to profiling tools
    ● to understand and tune OpenACC performance
    ● to help you understand a real application to start OpenACC-ing...

● The idea is to equip you with the knowledge to develop applications that run efficiently on parallel hybrid supercomputers
  ● not just on single GPUs

"Accelerating the Way to Better Science"

Cray XK6 supercomputer

● Node architecture:
  ● One AMD Series 6200 Interlagos CPU (16 cores)
    ● Cray XE6 (HECToR, Lindgren...) has two of these per node
  ● One Nvidia GPU
    ● Currently Fermi+
      ● 512 cores, 665 GFlop/s DP, 6GB memory
      ● Upgradable to Kepler K20 in Autumn 2012
        ● 1536 cores
  ● Cray Gemini interconnect
    ● shared between two nodes
    ● high bandwidth/low latency scalability

● Fully integrated/optimised/supported
  ● Tight integration of GPU and NIC drivers
Cray hybrids in future Top500

ORNL Titan: 200 cabinets of Cray XK6 (19k nodes, ≥15k with Kepler GPUs)

NCSA Blue Waters: 235 cabinets of Cray XE6 + 30 cabinets of Cray XK6

The Exascale is coming...

- Sustained performance milestones every 10 years...
  - 1000x the performance with 100x the PEs

(And they're all Crays)
Exascale, not exawatts

- Power is a big consideration in an exascale architecture
  - Jaguar (ORNL) draws 6MW to deliver 1PF
  - The US DoE demands 1EF from only 20MW (and $200M)
- A hybrid system is one way to reach this, e.g.
  - $10^5$ nodes (up from $10^4$ for Jaguar)
  - $10^4$ FPUs/node (up from 10 for Jaguar)
  - some full-featured cores for serial work
  - a lot more cutdown cores for parallel work
  - Instruction level parallelism will be needed
    - continues the SIMD trend SSE → AVX → ...
- This looks a lot like the current GPU accelerator model
  - manycore architecture, split into SIMT threadblocks
  - Complicated memory space/hierarchy (internal and PCIe)
- And this looks a lot like the old days
  - welcome back to vectorisation, we kept the compiler ready for you

The Exascale furrow is a hard one to plough...

"Seymour Cray, the pioneer of supercomputing, famously once asked if you would rather plough a field with two strong oxen or five-hundred-and-twelve chickens.

"Since then, the question has answered itself: power restrictions have driven CPU manufacturers away from “oxen” (powerful single-core devices) towards multi- and many-core “chickens”.

"An exascale supercomputer will take this a step further, connecting tens of thousands of many-core nodes.

"Application programmers face the challenge of harnessing the power of tens of millions of threads."

EPCC News, issue 70 (Autumn 2011)
EU FP7 Network: Collaborative Research into Exascale Software Tools and Applications

Consortium has
- Leading European HPC centres
  - EPCC, HLRS, CSC, PDC
- Hardware partner
  - Cray
- Tools providers
  - TUD (Vampir), Allinea (DDT)
- Codesign application owners, specialists
  - ABO, JYU, UCL, ECMWF, ECP, DLR

Codesign approach
- Focus on real problems in real applications
- 6 apps: GROMACS, OpenFOAM, IFS, Elmfire, Nek5000, HemeLB

CRESTA and its two partner projects are the first Exascale development projects funded by Europe

Accelerator programming

Why do we need a new GPU programming model?

Aren’t there enough ways already?
- CUDA (incl. NVIDIA CUDA-C & PGI CUDA-Fortran)
- OpenCL
- Stream
- hiCUDA...

All are quite low-level and closely coupled to the GPU
- User needs to rewrite kernels in specialist language:
  - Hard to write and debug
  - Hard to optimise for specific GPU
  - Hard to port to new accelerator
- Multiple versions of kernels in codebase
  - Hard to add new functionality
CUDA on Cray XK6

- If you work hard, you can get good parallel performance
- Ludwig Lattice Boltzmann code rewritten in CUDA
  - Reordered all the data structures (structs of arrays)
  - Pack halos on the GPU
  - Streams to overlap compute, PCIe comms, MPI halo swaps
- 10 cabinets of Cray XK6
  - 936 GPUs (nodes)
- Only 4% deviation from perfect weak scaling between 8 and 936 GPUs.

Directive-based programming

- Most scientific applications will not have this level of developer support (Ludwig was special research case)
- Directives provide high-level approach
  - Based on original source code (e.g. Fortran, C, C++)
  - Easier to maintain/port/extend code
  - Users with (for instance) OpenMP experience find it a familiar programming model
  - Compiler handles repetitive boilerplate code (cudaMalloc, cudaMemcpy,...)
  - Compiler handles default scheduling; user can step in with clauses where needed
    - Possible performance sacrifice
      - Important to quantify this
      - Can then tune the compiler
      - Small performance sacrifice is an acceptable trade-off for portability and productivity
        - Who handcodes in assembler these days?
- Two relevant performance comparisons:
  - How does the performance compare to CUDA?
  - Can I justify buying a GPU instead of another CPU?
Performance compared to CUDA

- Is there a performance gap relative to explicit low-level programming model? Typically 10-15%, sometimes none.
- Is the performance gap acceptable? Yes.
  - e.g. S3D comp_heat kernel (ORNL application readiness):

![Graph showing performance comparison]

Node-for-node performance comparison

- Does accelerated parallel application performance justify buying a GPU (Cray XK6)
  - rather than another CPU (Cray XE6)?
  - For many codes, yes.

![Graph showing performance comparison for Himeno Benchmark]
Structure of this course

- **Aims to lead you through the entire development process**
  - What is OpenACC?
  - How do I use it in a simple code?
  - How do I port a real-sized application?
  - Performance tuning and advanced topics
  - Case studies

- **It will assume you know**
  - A little bit about GPU architecture and programming
    - SMs, threadblocks and warps, coalescing
    - as covered in the CUDA part of this course

- **It will help if you know**
  - The basic idea behind OpenMP programming
    - but this is not essential