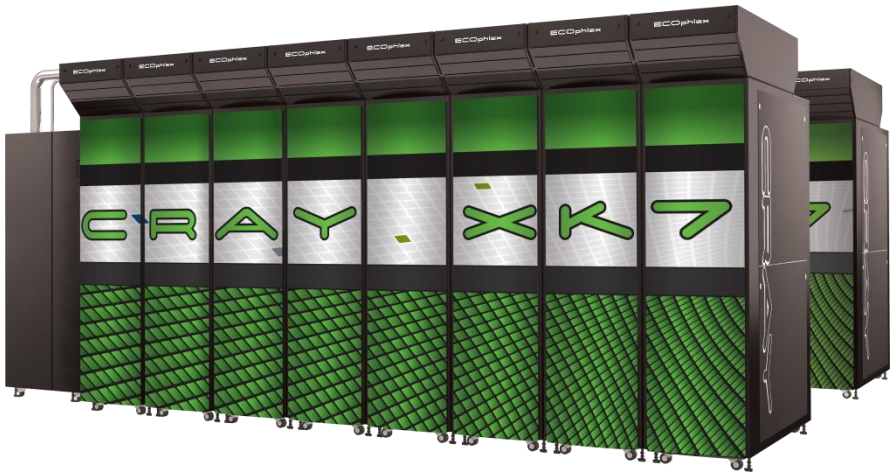
Practical: a sample code

Alistair Hart
Cray Exascale Research Initiative Europe



Aims

- The aim of this practical is to examine, compile and run a simple, pre-prepared OpenACC code

The sample code

- **The sample code**
  - Designed to demonstrate functionality
  - not interested in performance at this stage
- **Implements the simple example from the lectures**
  - A 3d array a is initialised
  - It's values are doubled and stored in a new array b
  - A checksum is calculated and compared with the expected result
- **These are implemented as 3 OpenACC kernels**

- **There are three versions of the code**
  - Version 00 has all 3 kernels in same main program.
    - There is no attempt to keep data on the GPU between the kernels.
  - Version 01 uses a data region to avoid data sloshing.
  - Version 02 has more complicated calltree
    - calls a subroutine that contains an OpenACC kernel.
    - This kernel also contains a function call.

Code versions and building them

- **There are versions for 4 different programming models**
  - C or Fortran, with
  - static or dynamic allocation of arrays
    - N.B. there is no version00 for dynamic arrays with C (see note in version01)
  - source filename based on these, e.g. first_example_Fstatic_v00.f90

- **Get your environment right**
  - make sure you have the right PrgEnv loaded (cray or pgi)
  - make sure you have loaded the correct compiler version module
  - make sure you have loaded module craype-accel-nvidia20

- **Build the code**
  - PrgEnv-cray:
    - ftn -hlist=a <Fortran source file>
    - cc -hlist=a <C source file>
  - PrgEnv-pgi:
    - ftn -Minfo=all <Fortran source file>
    - cc -Minfo=all <C source file>

Automation

- **You can do it all by hand if you wish, or use automation**
  - There's nothing magic being done here
- **Automated building:**
  - can just type: make VERSION=[00|01|02] [F|C][static|dynamic]
    - Makefile will echo commands it uses to build the code
    - automatically detects which PrgEnv you are using (uses PE_ENV env. var.)
      - remember to type "make clean" if you switch PrgEnv modules
- **Automated building and job submission**
  - type: bash build_submit.bash MYPE TARGET VERSION
    - MYPE should be cray or pgi
    - TARGET should be Fstatic or Fdynamic or Cstatic or Cdynamic
    - VERSION should be 00 or 01 or 02
  - This will:
    - load the correct modules using script ../XK_setup.bash
    - build the code using the Makefile
    - create directory: /lus/scratch/$USER/Cray_OpenACC_training/Practical1/TARGET_VERSION_DATE_TIME
    - write and submit a PBS jobscript
  - You can then cd to this directory and look at the output

What to check

- **Check correctness**
  - Did the code compile correctly?
  - Did the job execute?
  - Was the answer correct?

- **Next, understand what did the compiler did**
  - examine and understand the compiler feedback
    - CCE: open the .lst file
    - PGI: read the output to stdout
  - did it compile for the accelerator?
  - what data did it plan to move and when?
  - how were the loop iterations scheduled?

What actually ran?

- **Did we actually run on the accelerator?**
  - We can ask the runtime for some feedback
  - cd to run directory, edit jobscript, uncomment appropriate line
    - CCE: set CRAY_ACC_DEBUG to 1 (least detailed) to 3 (most detailed)
    - PGI: set ACC_NOTIFY
  - Resubmit the job: qsub <jobscript name>
  - Examine commentary (in the log file) and make sure you understand it

- **Profiling the code**
  - A quick way of profiling is to use the Nvidia compute profiler
    - CCE and PGI compile to PTX (as does nvcc), so this will work for all
  - Edit the jobscript and uncomment the profiling line
  - Resubmit the job
  - Examine the profile (in file cuda_profile_0.log)
    - Can change location with env. var. COMPUTE_PROFILE_LOG
  - This is a "blow-by-blow" account
    - Larger codes need a more aggregated report
    - We will cover profiling in more detail later

Further work

- **Choose a target and repeat this for all three versions**
- **Start with the Cray compiler**
- **Then either:**
  - repeat for a different programming model target, or
  - try the PGI compiler

Getting the examples

- **On raven:**
  - change to a directory where you want to work
    - either in your home directory or under /lus/scratch/$USER
  - type: tar zxvf ~tr99/cray_OpenACC_training.tgz
  - This creates a new directory ./Cray_OpenACC_training
    - please note the file LICENCE.txt
  - The codes for Practical 1 are in:
    - Cray_OpenACC_training/Practical1
    - There is a README file that summarises these slides
  - ~tr99/Slides/* contains PDFs of the slides supporting the practicals