Accelerate Your Science:
An Introduction to High Performance Computing
Lab 3: An Introduction to OpenMP

Guido Juckeland
Visiting Scholar
Technische Universität Dresden, Germany

July 8, 2014
Task 1: Approximate Pi

- Mathematically, we know that:

  \[ \pi = \int_0^1 \frac{4.0}{1 + x^2} \, dx \]

- We can approximate the integral as a sum of rectangles:

  \[ \pi \approx \sum_{i=0}^{N} F(x_i) \Delta x \]

  where each rectangle has width \( \Delta x \) and height \( F(x_i) \) at the middle of the interval \( i \).
Task 1: Approximate Pi (2)

- Use the provided `pi.c` serial program and parallelize it using OpenMP
- Consider two versions
  - Using workshare paradigms
  - Distributing the work manually to the threads
- Pay close attention to data sharing / private data
- In addition to a parallel construct, you might need the runtime library routines
  - `int omp_get_num_threads();`  // total number of threads
  - `int omp_get_thread_num();`  // id of my thread
  - `double omp_get_wtime();`  // timestamp in seconds
Hints about the environment

- OpenMP lab examples use a mechanism called **Makefiles**
- Work with any programming environment
- You need to tell the `make` system what options to use:
  ```
  cp linux_cray.def make.def
  ```
- Use other `def` files for other environments

- To build your code
  ```
  make pi
  ```
- To run your code use (from an interactive job)
  ```
  OMP_NUM_THREADS=32 aprun -d 32./pi
  ```
- To clean up your work
  ```
  make clean
  ```
Task 2: Doing Pi differently

- The area of the unit circle is $\pi$
- Using two random numbers for a position in x and y, one can count the number of hits inside the circle vs. the total number of tries
- The area of the surrounding square is known to be 4
- Hence $\pi$ can be approximated as:

$$\frac{\pi}{4} = \frac{n_{\text{hits}}}{n}$$
Task 2: Doing Pi differently (2)

- This approach is a Monte-Carlo-method
- Modify the provided `pi_mc.c` to use OpenMP