alpaka Parallel Programming – Online Tutorial
Lecture 20: Thread Parallelism in alpaka
Lesson 23: Computing $\pi$ – Part I
Lesson 23: Computing π – Part I

Computing π

• Focus of the next four lessons
• Good example for Thread parallelism
• Introduces parameter passing and memory management
• Initial algorithm: Find points in a circle
Points in a circle

- Task: Given a circle quarter with the radius $r$ and a set of $n$ randomly scattered points, find all points inside the circle quarter
- Approach:
  - Create a Grid with $n$ Threads
  - Each Thread evaluates a single point
Algorithm

- Using Pythagoras’ theorem, the distance $d$ from a point to the origin can be calculated:
  \[ d = \sqrt{x^2 + y^2} \]
- If $d \leq r$, return `true`, otherwise `false`
Lesson 23: Computing π – Part I

Kernel requirements

• For the computation we need:
  • The point coordinates:
    ```
    struct Points {
        float * x;
        float * y;
        bool * inside;
    };
    ```
  • The radius: float r;
  • How do we pass these to the kernel?
Lesson 23: Computing π – Part I

Passing parameters

- alpaka kernels accept three different parameter types:
  - The accelerator: `Acc const & acc` (required)
  - Pointers to memory buffers of any data type: `float * bufferA, MyDataType * bufferB`
  - Scalar values of trivially copyable types: `float scalar, struct Composed { int a; float b; };`
- Signature of the `PixelFinderKernel`’s `operator()`: ```
  template <typename Acc>
  ALPAKA_FN_ACC void operator()(Acc const & acc, // required
                                   Points points, // this struct contains memory buffers
                                   float r        // this is a scalar
                              ) const
```