alpaka Parallel Programming – Online Tutorial

Lecture 20: Thread Parallelism in alpaka

Lesson 24: Computing π – Part II

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

www.casus.science

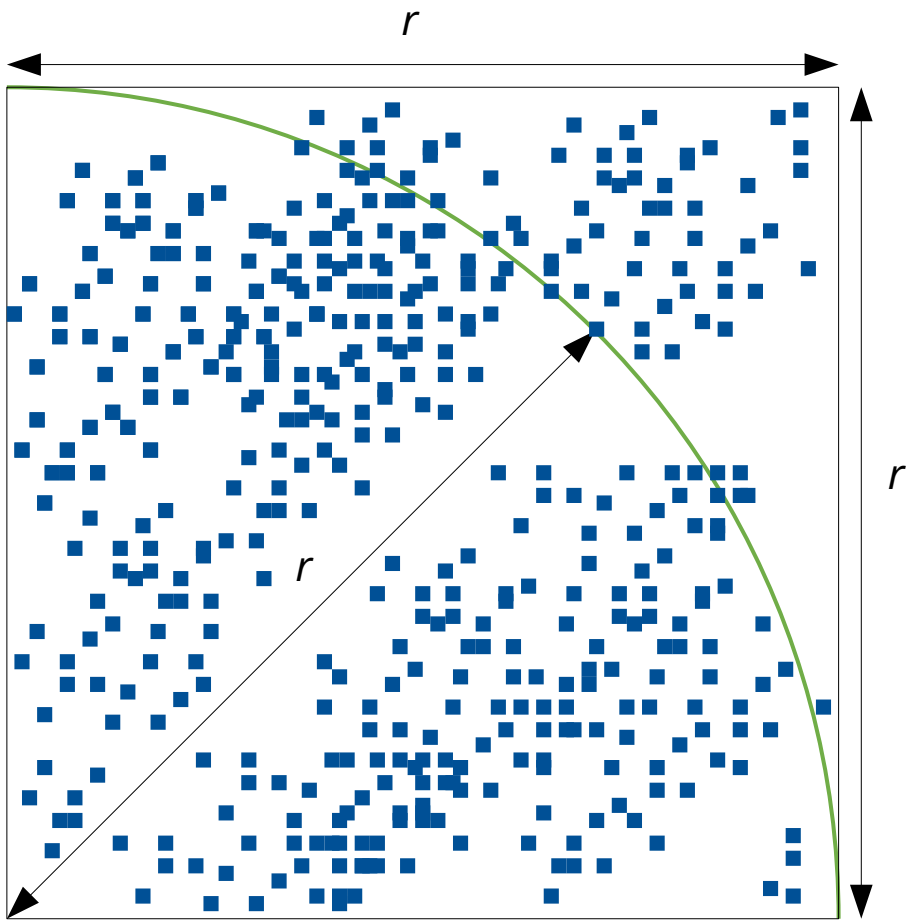
HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF
TECHNISCHE UNIVERSITÄT DRESDEN
CBG Max Planck Institute of Molecular Cell Biology and Genetics
UFZ HELMHOLTZ Centre for Environmental Research
Uniwersytet Wrocławski

SPONSORED BY THE
Federal Ministry of Education and Research
STAATSMINISTERIUM FÜR WISSENSCHAFT KULTUR UND TOURISMUS
Freistaat SACHSEN

Lesson 24: Computing π – Part II

Recap

- Introduced goal: Compute π in parallel
- Introduced initial algorithm: find points in circle
- Introduced kernel requirements
- Introduced parameter passing
- Next: Find the points!



Lesson 24: Computing π – Part II

Grid dimensionality

- No spatial relationship between points
- Points can be evaluated independently
- This makes a multi-dimensional grid unnecessary

```
struct PixelFinderKernel
{
    template <typename Acc>
    ALPAKA_FN_ACC void operator()(Acc const & acc, Points points, float r) const {

        using namespace alpaka;

        uint32_t gridThreadIdx = idx::getIdx<Grid, Threads>(acc)[0];
        /* … */
```

Lesson 24: Computing π – Part II

Accessing memory

- Iterating over a buffer works differently in alpaka
- `for` loop: One thread accesses elements sequentially
- Thread index: Threads access elements in parallel
- If required, you can mix both approaches!

```
// Using a for loop for buffer access
for(std::size_t i = 0; i < n; ++i)
{
    float x = points.x[i];
    float y = points.y[i];
}
```

```
// Using the thread index for buffer access
float x = points.x[gridThreadIdx];
float y = points.y[gridThreadIdx];
```

Lesson 24: Computing π – Part II

Computing the distance

- Use Pythagoras' theorem for computing the distance
- Use `math::sqrt()` for computing the square root
  - Requires the `acc` parameter!

```
        /* … */
        float d = math::sqrt(acc, x * x + y * y);

        bool isInside = (d <= r);

        points.inside[gridThreadIdx] = isInside;
    }
};
```

Lesson 23: Computing π – Part I

CASUS
CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

The complete Kernel

```
struct PixelFinderKernel
{
    template <typename Acc>
    ALPAKA_FN_ACC void operator()(Acc const & acc, Points points, float r) const {

        uint32_t gridThreadIdx = idx::getIdx<Grid, Threads>(acc)[0];

        float x = points.x[gridThreadIdx];
        float y = points.y[gridThreadIdx];
        float d = math::sqrt(acc, x * x + y * y);

        bool isInside = (d <= r);

        points.inside[gridThreadIdx] = isInside;
    }
};
```

CASUS

CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

www.casus.science

HZDR HELMHOLTZ ZENTRUM DRESDEN ROSSENDORF

TECHNISCHE UNIVERSITÄT DRESDEN

CBG Max Planck Institute of Molecular Cell Biology and Genetics

UFZ HELMHOLTZ Centre for Environmental Research

Uniwersytet Wrocławski

SPONSORED BY THE

Federal Ministry of Education and Research

STAATSMINISTERIUM FÜR WISSENSCHAFT KULTUR UND TOURISMUS

Freistaat SACHSEN