alpaka Parallel Programming – Online Tutorial

Lecture 20 – Thread Parallelism in alpaka

Lesson 26: Computing π – Part IV

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 26: Computing π – Part IV

Recap

- Introduced parameter passing
- Introduced mathematical functions
- Introduced memory management
- Now: compute π

Lesson 26: Computing π – Part IV

Approach

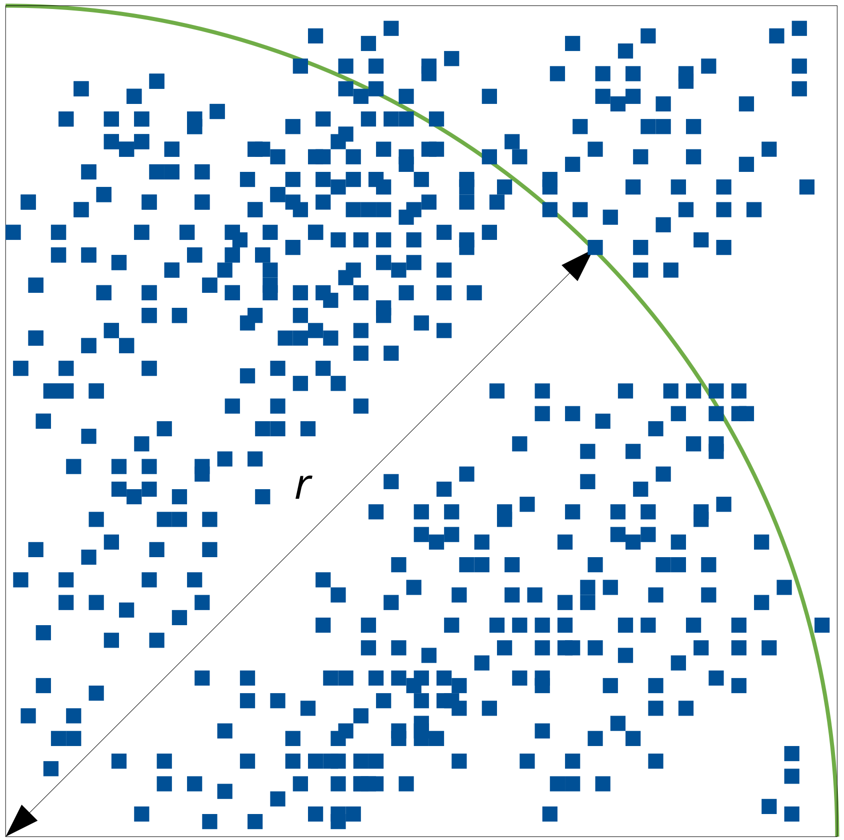
- We will use the formula for the area of a circle quarter:

  $$A = \frac{\pi \cdot r^2}{4}$$

- The number of points inside the circle (*P*) can be used to approximate *A:*

  $$\frac{P}{n} \approx \frac{A}{r^2} = \frac{\pi}{4} \rightarrow \pi \approx \frac{4P}{n}$$

- The `PixelFinderKernel` does the counting on the Device, integration is done by the Host.



Lesson 26: Computing π – Part IV

CASUS
CENTER FOR ADVANCED
SYSTEMS UNDERSTANDING

Kernel execution and memory transfer

- We will measure the execution time:

```
auto start = std::chrono::steady_clock::now();
```

- Execute the kernel using `alpaka::kernel::exec()`:

```
PixelFinderKernel pixelFinderKernel;
auto taskRunKernel = kernel::createTaskKernel<Acc>(workDiv, pixelFinderKernel,
                                                   pointsAcc, r);
queue::enqueue(queue, taskRunKernel);
```

- Copy back the results and synchronize:

```
mem::view::copy(devQueue, insideBufferHost, insideBufferAcc, extents);
alpaka::wait::wait(queue);
```

Lesson 26: Computing π – Part IV

Integration

- First, determine *P*:

```
uint64_t P = 0;
for(std::size_t i = 0; i < n; ++i)
{
    if(pointsHost.inside[i])
        ++P;
}
```

- Then, divide by the radius to approximate π:

```
float pi = (4.f * P) / n;
```

- Measure the execution time:

```
auto end = std::chrono::steady_clock::now();
```

Lesson 26: Computing π – Part IV

CASUS CENTER FOR ADVANCED SYSTEMS UNDERSTANDING

Aftermath

- Print out π and execution time:

```
std::chrono::duration<double, std::milli> duration = end – start;
std::cout << "Computed pi is " << pi << "\n";
std::cout << "Execution time: " << duration.count() << "ms" << std::endl;
```

- Homework #1: Play around with *n*. How does this affect the precision of π and the execution time?
- Homework #2: Implement the kernel in a more generic way, so that it works for any number of threads, blocks and grids.
  - The workload has to be distributed between all threads in the grid.
  - It requires to have a loop over points inside the kernel. A sample is given in a Q&A answer from Tuesday.

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