Radically Simplified GPU Parallelization: The Alea Dataflow Programming Model

Philipp Kramer, Luc Bläser
Institute for Software, HSR Rapperswil
Daniel Egloff
QuantAlea, Zurich

Funded by Swiss Commission of Technology and Innovation,
Project No 16130.2 PFES-ES

HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL
FHO Fachhochschule Ostschweiz

INSTITUT FÜR SOFTWARE

QuantAlea

GTC 2015
20 Mar 2015

GPU Parallelization Is Tough

- Massive Parallel Power
- High obstacles
  - Vector parallel algorithms needed
  - Machine-centric programming models (CUDA, OpenCL)
  - Invocation and data management logic

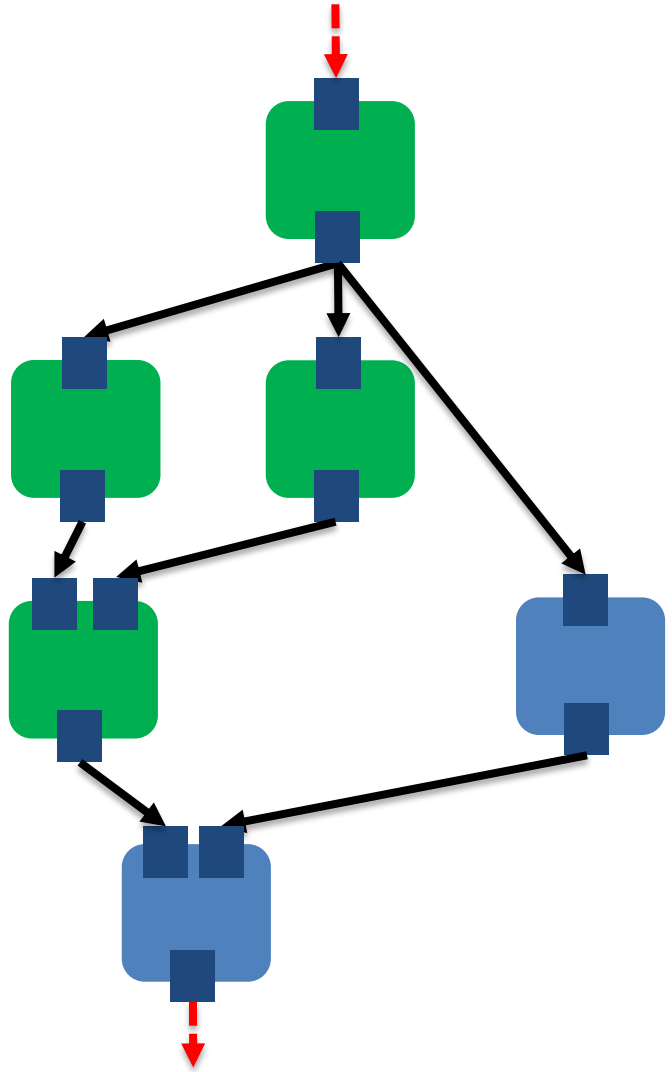
5760 Cores

Simple GPU Programming

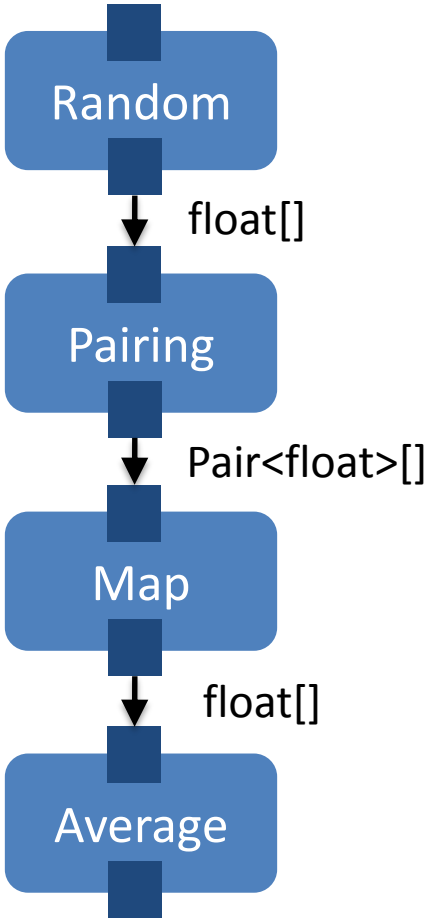
- GPU parallel programming for (almost) everyone
  - No GPU experience required
  - Fast development
  - Good performance
- On the basis of .NET
  - Available for C#, F#, VB etc.

Alea Dataflow Programming Model

- Dataflow
  - Graph of operations
  - Data propagated through graph
- Reactive
  - Feed input in arbitrary intervals
  - Listen for asynchronous output



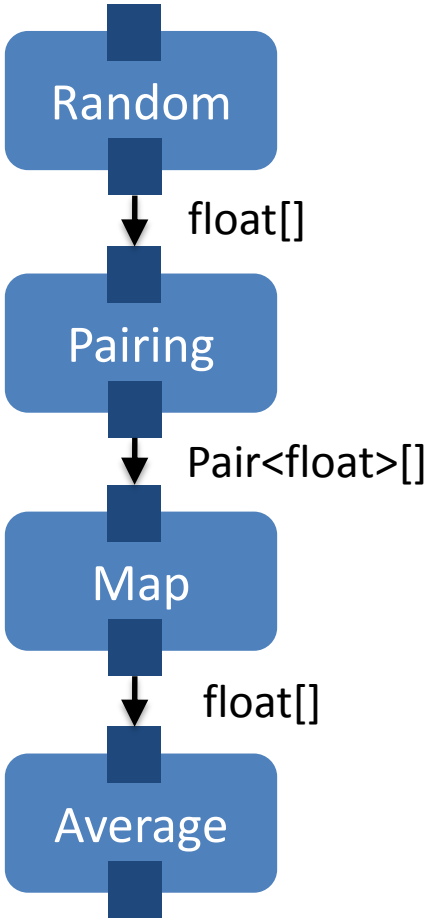
Graph of Operations




```
var randoms = new Random<float>(0, 1);
var coordinates = new Pairing<float>();
var inUnitCircle = new Map<Pair<float>, float>
  (p => p.Left * p.Left +
        p.Right * p.Right <= 1
        ? 1f : 0f);
var average = new Average<float>();
```

```
randoms.Output.ConnectTo(coordinates.Input);
coordinates.Output.ConnectTo(inUnitCircle.Input);
inUnitCircle.Output.ConnectTo(average.Input);
```

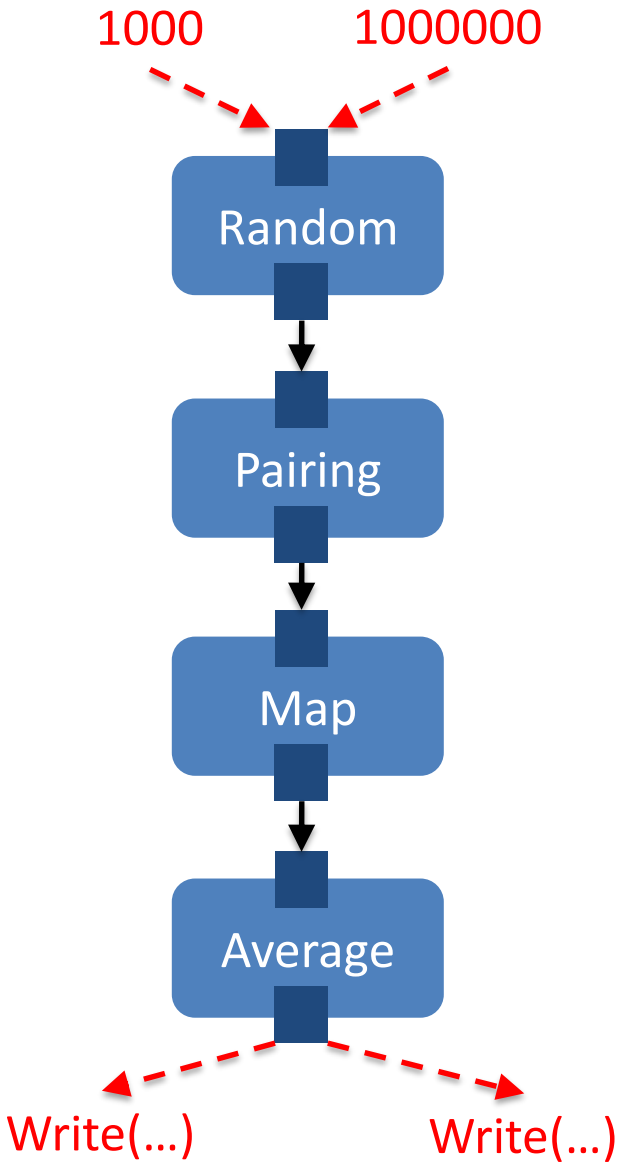
Graph of Operations




```
var randoms = new Random<float>(0, 1);

randoms
   .Pair()
   .Map(p => p.Left * p.Left +
             p.Right * p.Right <= 1
             ? 1f : 0f)
   .Average
```

Dataflow



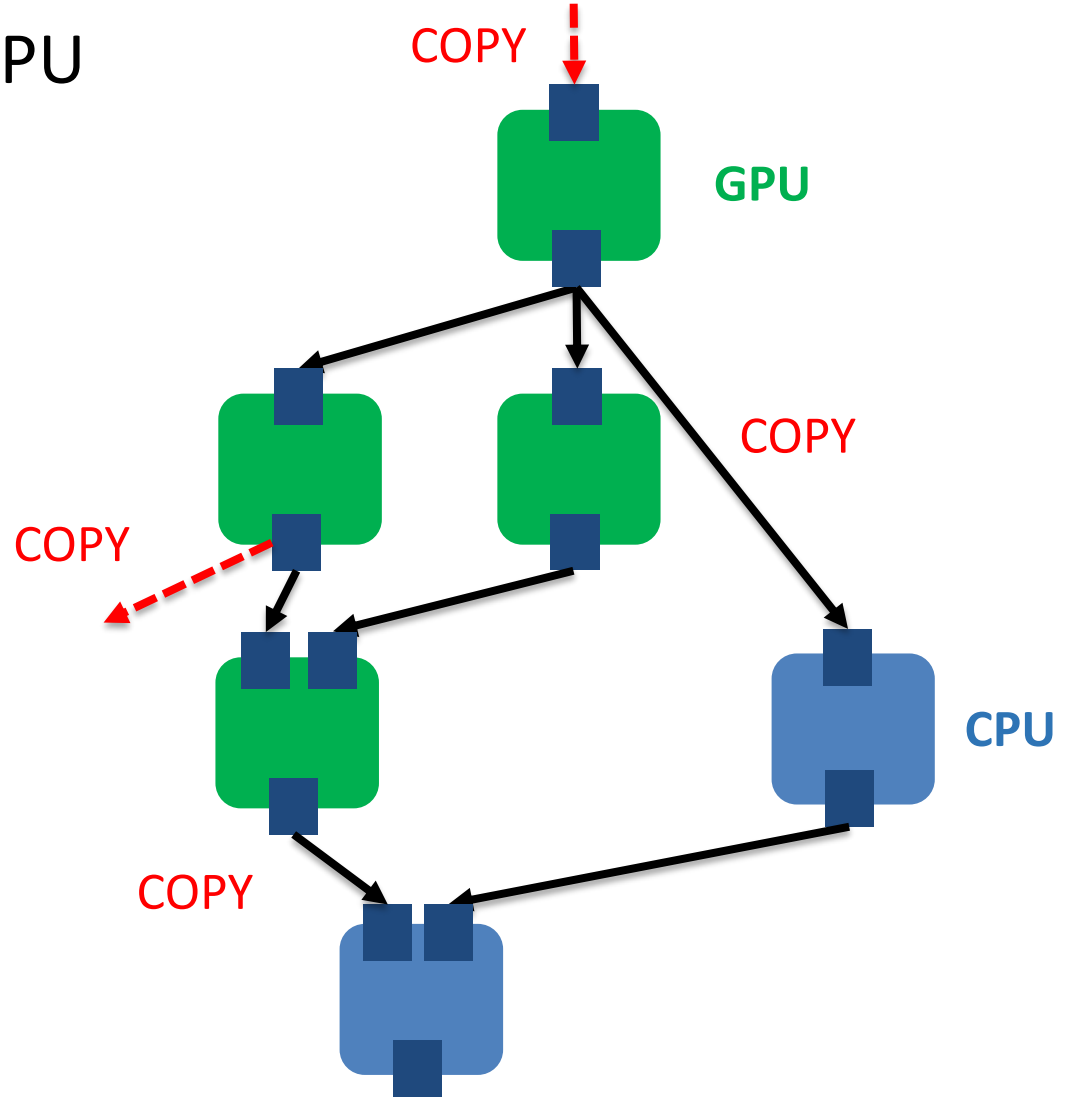
- Send data to input port
- Receive from output port
- All asynchronous

```
average.Output.OnReceive(x ->
        Console.WriteLine(4 * x));

random.Input.Send(1000);
random.Input.Send(1000000);
```

Runtime System behind the Scenes

- Operations implement GPU and/or CPU
- Automatic memory copying
  - only when needed
- GPU garbage collection
- Scheduling with .NET TPL



Performance

- GeForce GTX Titan Black (2880 cores) vs. CPU (4 Core Intel Xeon E5-2609 2.4GHz)
- MC Option Pricing Case

Configuration	Speedup
32 options, 16k paths/it, 30 days	6
32 options, 32k paths/it, 90 days	18
32 options, 128k paths/it, 360 days	30

- Training Phase of Neural Network Case (MNIST data)

Configuration	Speedup
60k images, 750 size, 30 hidden neurons	1
60k images, 3k size, 200 hidden neurons	20
60k images, 3k size, 600 hidden neurons	30

Conclusions

- Simple GPU parallelization in .NET
  - Fast development with generic prefabricated operations
  - Possibility to define custom operations
- Efficient runtime system
  - Automatic memory management
  - Low overhead compared to CUDA C

Further Information

- www.quantalea.com

Prof. Dr. Luc Bläser
HSR Hochschule für Technik Rapperswil
Institute for Software
lblaeser@hsr.ch
http://concurrency.ch

Dr. Daniel Egloff
QuantAlea GmbH

Switzerland
daniel.egloff@quantalea.net
www.quantalea.com

HSR HOCHSCHULE FÜR TECHNIK RAPPERSWIL
FHO Fachhochschule Ostschweiz

INSTITUT FÜR SOFTWARE

QuantAlea