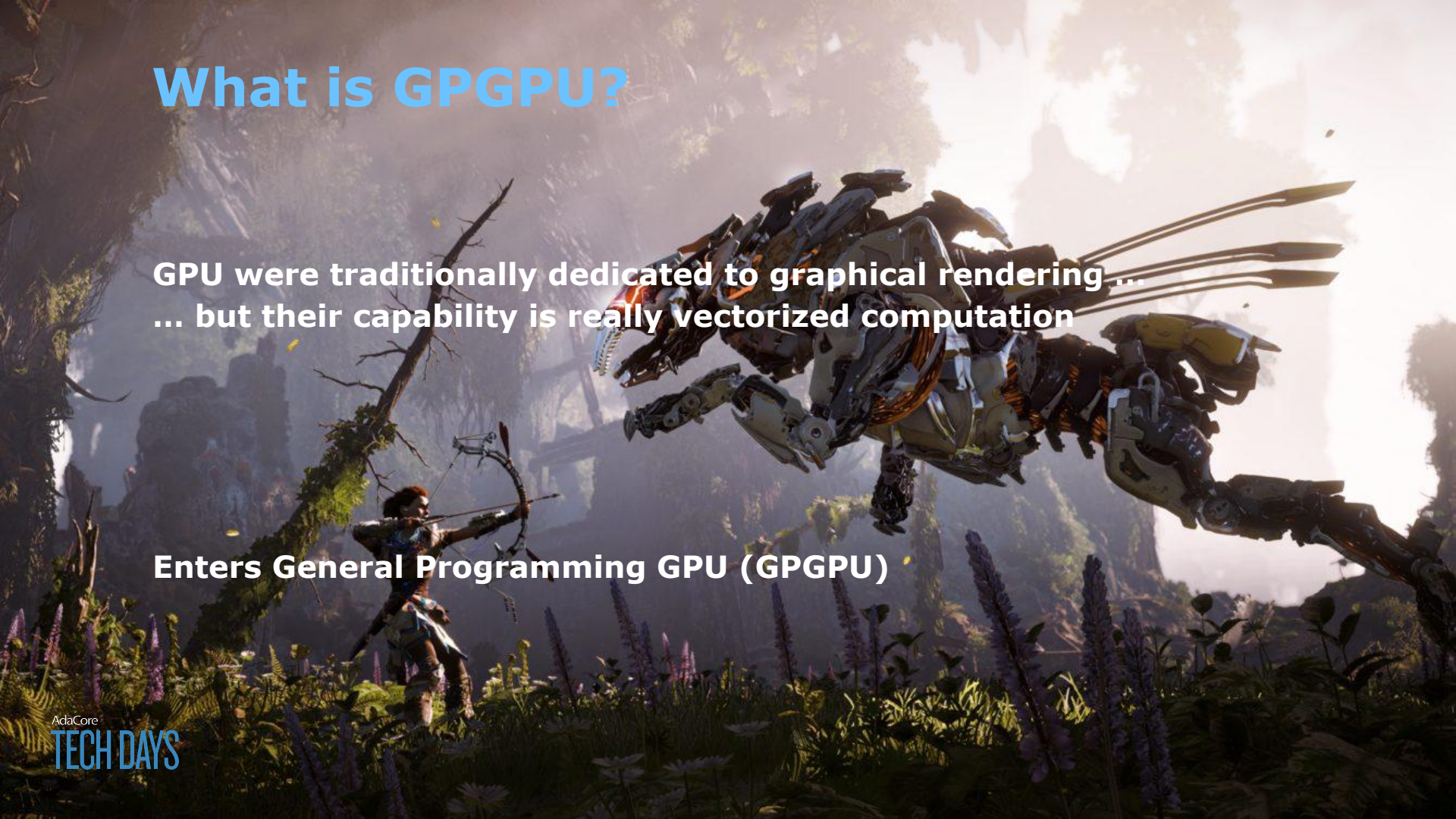

AdaCore
TECH DAYS

General Programming on Graphical Processing Units

Quentin Ochem
October 4th, 2018

What is GPGPU?



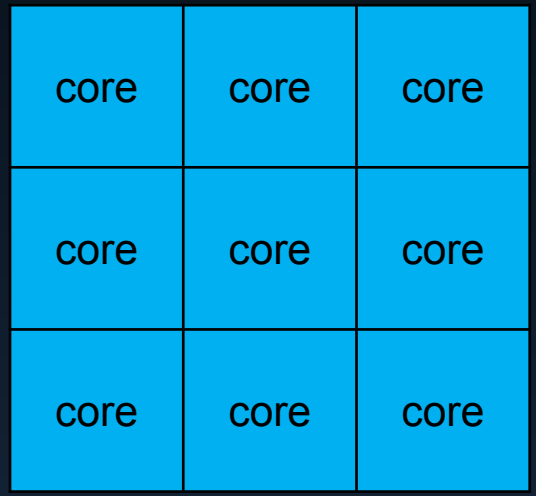
GPU were traditionally dedicated to graphical rendering …
… but their capability is really vectorized computation

Enters General Programming GPU (GPGPU)

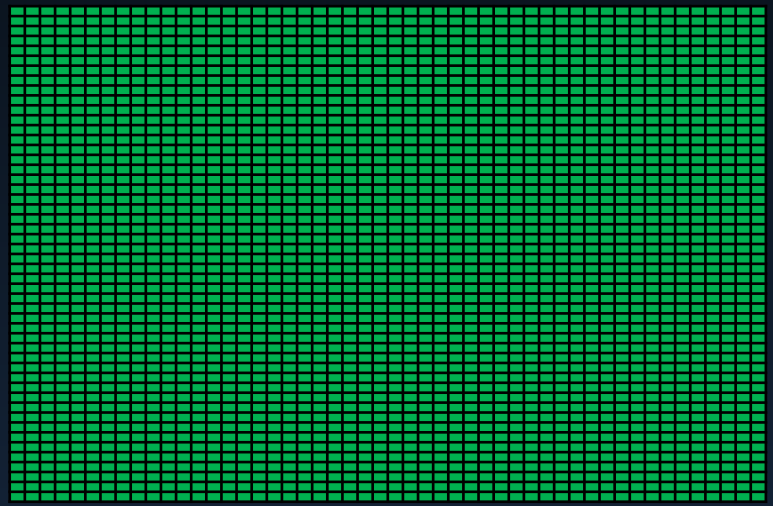
GPGPU Programming Paradigm

Debug?

Optimize data transfer?



Offload computations

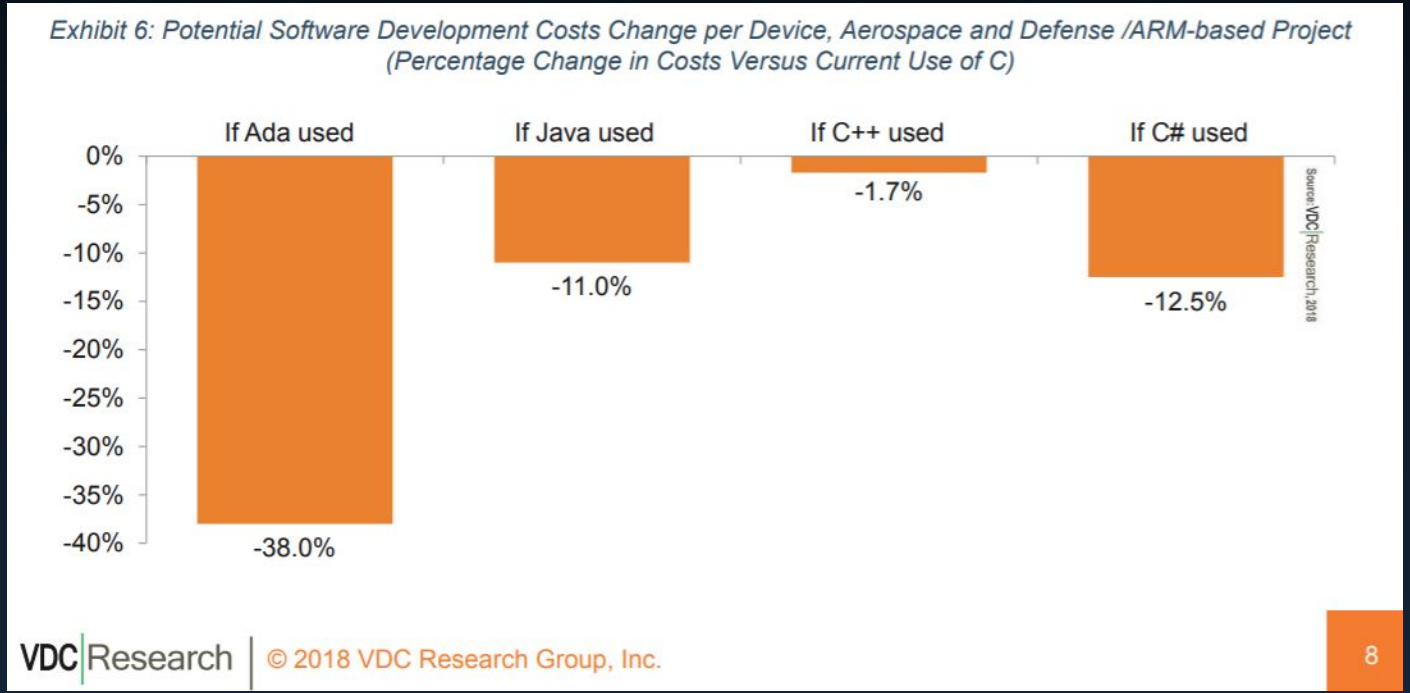


Refactor parallel algorithms?

Avoid data races?

How to optimize occupancy

Why do we care about Ada? (1/2)



AdaCore TECH DAYS

Source: https://www.adacore.com/uploads/techPapers/Controlling-Costs-with-Software-Language-Choice-AdaCore-VDC-WP.PDF

Why do we care about Ada (2/2)

- Signal processing
- Machine learning
- Monte-carlo simulation
- Trajectory prediction
- Cryptography
- Image processing
- Physical simulation

- … and much more!

Available Hardware

Desktop & Server

NVIDIA GeForce / Tesla / Quadro

AMD Radeon

Intel HD

Embedded

NVIDIA Tegra

ARM Mali

Qualcomm Adreno

IMG Power VR

Freescale Vivante

Ada Support

Three options

Interfacing with existing libraries

“Ada-ing” existing languages

Ada 2020

Interfacing existing libraries

Already possible and straightforward effort

“gcc –fdump-ada-specs” will provide a first binding of C to Ada

We could provide “thick” bindings to e.g. Ada.Numerics matrix operations

“Ada-ing” existing languages

CUDA – kernel-based language specific to NVIDIA

OpenCL – portable version of CUDA

OpenACC – integrated language marking parallel loops

CUDA Example (Device code)

```
procedure Test_Cuda
   (A : out Float_Array; B, C : Float_Array)
   with Export => True, Convention => C;
pragma CUDA_Kernel (Test_Cuda);

procedure Test_Cuda
   (A : Float_Array; B, C : Float_Array)
is
begin
   A (CUDA_Get_Thread_X) := B (CUDA_Get_Thread_X) + C (CUDA_Get_Thread_X);
end Test_cuda;
```

CUDA Example (Host code)

```
   A, B, C : Float_Array;
begin
   -- initialization of B and C
   -- CUDA specific setup

   pragma CUDA_Kernel_Call (Grid'(1, 1, 1), Block'(8, 8, 8));
   My_Kernel (A, B, C);

   -- usage of A
```

OpenCL example

-Similar to CUDA in principle

-Requires more code on the host code (no call conventions)

OpenACC example (Device & Host)

```
procedure Test_OpenACC is
   A, B, C : Float_Array;
begin
   -- initialization of B and C

   for I in A'Range loop
      pragma Acc_Parallel;
      A (I) := B (I) + C (I);
   end loop;
end Test_OpenACC;
```

Ada 2020

```
procedure Test_Ada2020 is
 A, B, C : Float_Array;
begin
   -- initialization of B and C

    parallel for I in A'Range loop
       A (I) := B (I) + C (I);
    end loop;
end Test_Ada2020;
```

Lots of other language considerations

- Identification of memory layout (per thread, per block, global)
- Thread allocation specification
- Reduction (ability to aggregate results through operators e.g. sum or concatenation)
- Containers
- Mutual exclusion
- ...

A word on SPARK

```
   X_Size : 1000;
   Y_Size : 10;

   Data : array (1 .. X_Size * Y_Size)  of Integer;
begin
   for X in 1 .. X_Size loop
      for Y in 1 .. Y_Size loop
         Data (X + Y_Size * Y) := Compute (X, Y);
      end loop;
   end loop;
```

{X = 100, Y = 1}, X + Y * Y_Size = 100 + 10 = **110**
{X = 10, Y = 10}, X + Y * Y_Size = 10 + 100 = **110**

Next Steps

AdaCore spent 1 year to run various studies and experiments

Finalizing an OpenACC proof of concept on GCC

About to start an OpenCL proof of concept on CCG

If you want to give us feedback or register to try technology, contact us on info@adacore.com