LINMA2710 - Scientific Computing
Graphics processing unit (GPU)

P.-A. Absil and B. Legat

☐ Full Width Mode ☐ Present Mode

Table of Contents

Sources

- OpenCL.jl
- HandsOnOpenCL
- Optimizing Parallel Reduction in CUDA
- Parallel Computation Patterns (Reduction)
- Profiling, debugging and optimization
- How to use TAU for Performance Analysis

Introduction

Context

Error message from Downloads

> RequestError: Could not resolve host: ddz4ak4pa3d19.cloudfront.net while requesting https://ddz4ak4pa3d19.cloudfront.net/cache/d3/a3/d3a36ce594d73649a043288f18a8896b.jpg

Stack trace

Here is what happened, the most recent locations are first:

1. anonymous function(easy::Downloads.Curl.Easy) ...show types...
   from **Downloads** → *Downloads.jl:452*
2. **with_handle**(f::Downloads.var"#9#19"{…}, handle::Downloads.Curl.Easy) ...show types...
   from **Downloads** → *Curl.jl:105*
3. anonymous function
   from *Downloads.jl:363*
4. **arg_write**(f::Downloads.var"#8#18"{…}, arg::IOStream) ...show types...
   from **ArgTools** → *ArgTools.jl:134*
5. anonymous function
   from *Downloads.jl:362*
6. arg_read
   from *ArgTools.jl:76*
7. **#request#5**(url::String; input::Nothing, output::IOStream, method::Nothing, headers::Vector{…}, timeout::Float64, progress::Nothing, verbose::Bool, debug::Nothing, throw::Bool, downloader::Nothing, interrupt::Nothing) ...show types...
   from **Downloads** → *Downloads.jl:361*
8. request

from Downloads.jl:328

9. anonymous function
   from Downloads.jl:259
10. **#open#465**(f::Downloads.var"#3#4"{…}, args::String; kwargs::@Kwargs{…}) ...show types...
    from **julia** → *io.jl:410*
11. open_nolock
    from *ArgTools.jl:35*
12. **arg_write**(f::Function, arg::String)
    from **ArgTools** → *ArgTools.jl:103*
13. #download#2
    from *Downloads.jl:258*
14. download
    from *Downloads.jl:247*
15. **#RobustLocalResource#27**(url::String, path::String, html_attributes::Pair{…}; cache::Bool) ...show types...
    from **PlutoTeachingTools** → *robustlocalresource.jl:20*
16. RobustLocalResource
    from *robustlocalresource.jl:7*
17. #save_image#19
    from *html.jl:39*
18. save_image
    from *html.jl:37*
19. #img#20
    from *html.jl:43*
20. img
    from *html.jl:42*
21. **#img#21**(file::String, args::Pair{…}; kws::@Kwargs{}) ...show types...
    from **MyUtils** → *html.jl:49*
22. **img**(file::String, args::Pair{…}) ...show types...
    from **MyUtils** → *html.jl:47*
23. from This cell: *line 1*

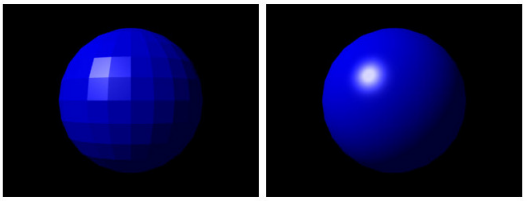
```
1 hbox([
2  md"""
```

```
hbox([
    md"""
* Most *dedicated* GPUs produced by
$(img("https://upload.wikimedia.org/wikipedia/commons/a/a4/NVIDIA_logo.svg",
:height => "15pt")) and
$(img("https://upload.wikimedia.org/wikipedia/commons/7/7c/AMD_Logo.svg", :height
=> "15pt"))
* *Integrated* GPUs by
$(img("https://upload.wikimedia.org/wikipedia/commons/6/6a/Intel_logo_%282020%2C_dar
k_blue%29.svg", :height => "15pt")) used in laptops to reduce power consumption
* Designed for 3D rendering through ones of the APIs :
$(img("https://upload.wikimedia.org/wikipedia/commons/7/7f/Microsoft-DirectX-Logo-
wordmark.svg", :height => "20pt")),
$(img("https://upload.wikimedia.org/wikipedia/commons/2/21/OpenGL_logo.svg",
:height => "20pt")),
$(img("https://upload.wikimedia.org/wikipedia/commons/2/25/WebGL_Logo.svg", :height
=> "20pt")),
$(img("https://upload.wikimedia.org/wikipedia/commons/2/2f/WebGPU_logo.svg",
:height => "25pt")),
$(img("https://upload.wikimedia.org/wikipedia/commons/f/fe/Vulkan_logo.svg",
:height => "20pt")) or Apple's Metal
$(img("https://upload.wikimedia.org/wikipedia/commons/8/8d/Metal_3_Logo.png",
:height => "20pt"))
* Illustration on the right is from [Charge's film]
(https://studio.blender.org/blog/charge-poster/?utm_medium=homepage), it shows how
3D modeling works.
""",
    img("https://ddz4ak4pa3d19.cloudfront.net/cache/d3/a3/d3a36ce594d73649a043288f18
    a8896b.jpg", :width => "120"),
])
```

Reading hidden

General-Purpose computing on GPU (GPGPU)

Also known as *compute shader* as they abuse the programmable shading of GPUs by treating the data as texture maps.



Error message from Main

```
RequestError: HTTP/2 429 while requesting
https://upload.wikimedia.org/wikipedia/commons/1/12/SYCL_logo.svg
```

Stack trace

Here is what happened, the most recent locations are first:

1. anonymous function
   from *Downloads.jl:271*
2. **#open#465**(f::Downloads.var"#3#4"{…}, args::String; kwargs::@Kwargs{…}) ...show types...
   from **julia** → *io.jl:410*
3. open_nolock
   from *ArgTools.jl:35*
4. **arg_write**(f::Function, arg::String)
   from **ArgTools** → *ArgTools.jl:103*
5. #download#2
   from *Downloads.jl:258*
6. download
   from *Downloads.jl:247*
7. **RobustLocalResource**(::String, ::String; cache::Bool)
   from **PlutoTeachingTools** → *robustlocalresource.jl:20*
8. RobustLocalResource
   from *robustlocalresource.jl:7*
9. #save_image#19
   from *html.jl:39*
10. save_image
    from *html.jl:37*
11. #img#20
    from *html.jl:43*
12. img
    from *html.jl:42*
13. **img**(::String; kws::@Kwargs{})
    from **MyUtils** → *html.jl:49*
14. **img**(::String)
    from **MyUtils** → *html.jl:47*
15. from This cell: *line 1*

```
1 grid([
```

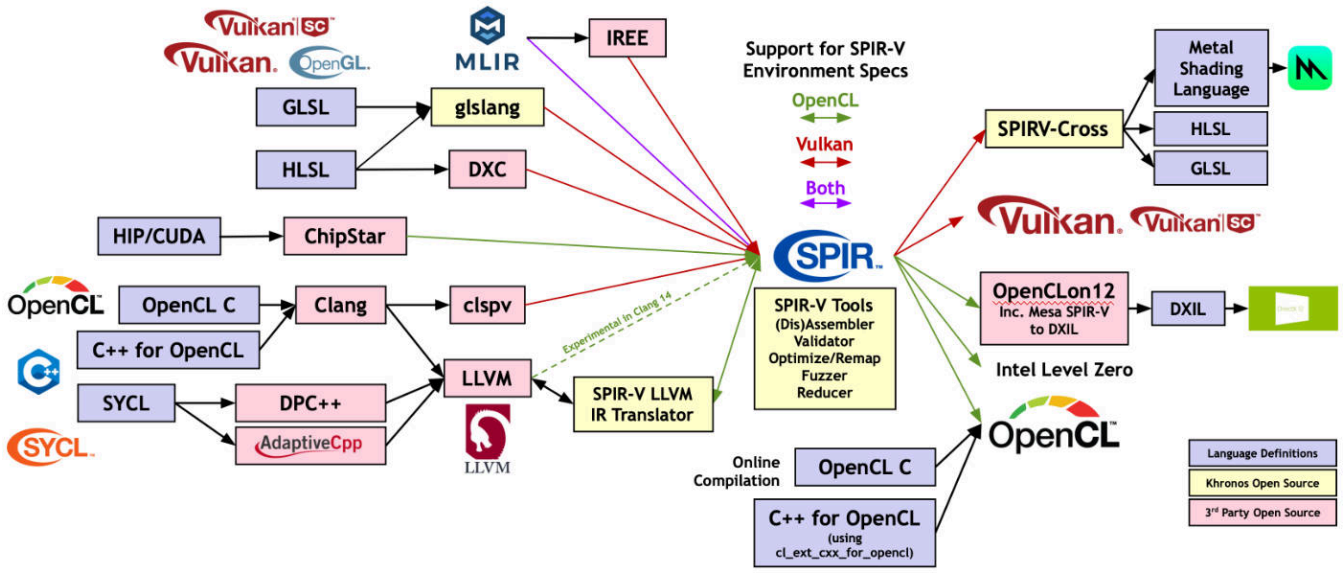
```
md"Hardware-specific" img("https://upload.wikimedia.org/wikipedia/commons/b/b9/Nvidia_CUDA_Logo.jpg", :height => "70pt") img("https://upload.wikimedia.org/wikipedia/commons/7/7b/ROCm_logo.png", :height => "60pt") md"""` ` $(img("https://upload.wikimedia.org/wikipedia/commons/6/6a/Intel_logo_%282020%2C_dark_blue%29.svg", :height => "30pt")) $(img("https://upload.wikimedia.org/wikipedia/en/f/fa/OneAPI-rgb-3000.png", :height => "60pt"))"""
```

```
grid([
    md"Hardware-specific"
    img("https://upload.wikimedia.org/wikipedia/commons/b/b9/Nvidia_CUDA_Logo.jpg",
    :height => "70pt")
    img("https://upload.wikimedia.org/wikipedia/commons/7/7b/ROCm_logo.png",
    :height => "60pt") md"""` `
    $(img("https://upload.wikimedia.org/wikipedia/commons/6/6a/Intel_logo_%282020%2C
    _dark_blue%29.svg", :height => "30pt"))
    $(img("https://upload.wikimedia.org/wikipedia/en/f/fa/OneAPI-rgb-3000.png",
    :height => "60pt"))"""
    md"Common interface"
    img("https://upload.wikimedia.org/wikipedia/commons/4/4d/OpenCL_logo.svg")
    img("https://upload.wikimedia.org/wikipedia/commons/1/12/SYCL_logo.svg")
    img("https://d29g4g2dyqv443.cloudfront.net/sites/default/files/akamai/designworks/blog1/OpenACC-logo.png", :height => "50pt")
])
```

Reading hidden

Standard Portable Intermediate Representation (SPIR)

Similar to LLVM IR : Intermediate representation for accelerated computation.



Hierarchy

Error message from Main

```
RequestError: HTTP/2 429 while requesting
https://upload.wikimedia.org/wikipedia/de/9/96/Platform_architecture_2009-11-08.svg
```

Stack trace

Here is what happened, the most recent locations are first:

1. anonymous function
   from *Downloads.jl:271*
2. **#open#465**(f::Downloads.var"#3#4"{…}, args::String; kwargs::@Kwargs{…}) ...show types...
   from **julia** → *io.jl:410*
3. open_nolock
   from *ArgTools.jl:35*
4. **arg_write**(f::Function, arg::String)
   from **ArgTools** → *ArgTools.jl:103*
5. #download#2
   from *Downloads.jl:258*

6. download
   from Downloads.jl:247
7. #RobustLocalResource#27(url::String, path::String, html_attributes::Pair{…}; cache::Bool) ...show types...
   from PlutoTeachingTools → robustlocalresource.jl:20
8. RobustLocalResource
   from robustlocalresource.jl:7
9. #save_image#19
   from html.jl:39
10. save_image
    from html.jl:37
11. #img#20
    from html.jl:43
12. img
    from html.jl:42
13. #img#21(file::String, args::Pair{…}; kws::@Kwargs{}) ...show types...
    from MyUtils → html.jl:49
14. img(file::String, args::Pair{…}) ...show types...
    from MyUtils → html.jl:47
15. from This cell: line 1

```
1 hbox([
2  md"""
```

```
hbox([
    md"""
* CPUs:
   - All CPUs part of same device
   - 1 Compute Unit per core
   - Number of processing elements equal to SIMD width
* GPUs:
   - One device per GPU
""",
    img("https://upload.wikimedia.org/wikipedia/de/9/96/Platform_architecture_2009-11-08.svg", :width => "400pt"),
])
```

👀 Reading hidden code

compute device	compute unit	processing element
get_global_id	get_group_id	get_local_id
get_global_size	get_num_groups	get_local_size

Memory

Error message from Main

```
RequestError: HTTP/2 429 while requesting
https://upload.wikimedia.org/wikipedia/de/d/d1/OpenCL_Memory_model.svg
```

Stack trace

Here is what happened, the most recent locations are first:

1. anonymous function
   from *Downloads.jl:271*
2. **#open#465**(f::Downloads.var"#3#4"{…}, args::String; kwargs::@Kwargs{…}) ...show types...
   from **julia** → *io.jl:410*
3. open_nolock
   from *ArgTools.jl:35*
4. **arg_write**(f::Function, arg::String)
   from **ArgTools** → *ArgTools.jl:103*
5. #download#2
   from *Downloads.jl:258*
6. download
   from *Downloads.jl:247*
7. **RobustLocalResource**(::String, ::String; cache::Bool)
   from **PlutoTeachingTools** → *robustlocalresource.jl:20*
8. RobustLocalResource
   from *robustlocalresource.jl:7*
9. #save_image#19

```
    from html.jl:39
10. save_image
    from html.jl:37
11. #img#20
    from html.jl:43
12. img
    from html.jl:42
13. img(::String; kws::@Kwargs{})
    from MyUtils → html.jl:49
14. img
    from html.jl:47
15. from This cell: line 1
```

```
1 img("https://upload.wikimedia.org/wikipedia/de/d/d1/OpenCL_Memory_model.svg")
```

```
1 img("https://upload.wikimedia.org/wikipedia/de/d/d1/OpenCL_Memory_model.svg")
```

👀 *Reading hidden*

OpenCL Platforms and Devices

- Platforms are OpenGL implementations, listed in `/etc/OpenCL/vendors`
- Devices are actual CPUs/GPUs
- ICD allows to change platform at runtime

```
1 OpenCL.versioninfo()
```

```
OpenCL.jl version 0.10.2

Toolchain:
 - Julia v1.11.6
 - OpenCL_jll v2024.5.8+1

Available platforms: 1
 - Portable Computing Language
   OpenCL 3.0, PoCL 7.0  Linux, Release, RELOC, SPIR-V, LLVM 20.1.2jl, SLEEF,
DISTRO, POCL_DEBUG
   · cpu-haswell-AMD EPYC 7763 64-Core Processor (usm, fp64, il)
```

See also `clinfo` command line tool and `examples/OpenCL/common/device_info.c`.

Tip

▶ **tl;dr To refresh the list of platforms, you need to quit Julia and open a new session**

Important stats

- Platform
  - name: Portable Computing Language
  - profile: FULL_PROFILE
  - vendor: The pocl project
  - version: PoCL 7.0 Linux, Release, RELOC, SPIR-V, LLVM 20.1.2jl, SLEEF, DISTRO, POCL_DEBUG
- Device
  - name: cpu-haswell-AMD EPYC 7763 64-Core Processor
  - type: cpu

clGetDeviceInfo	Value
CL_DEVICE_GLOBAL_MEM_SIZE	11.62 GiB
CL_DEVICE_MAX_COMPUTE_UNITS	4
CL_DEVICE_LOCAL_MEM_SIZE	512.00 KiB
CL_DEVICE_MAX_WORK_GROUP_SIZE	4096
CL_DEVICE_NATIVE_VECTOR_WIDTH_HALF	0
CL_DEVICE_NATIVE_VECTOR_WIDTH_FLOAT	8
CL_DEVICE_NATIVE_VECTOR_WIDTH_DOUBLE	4
CL_DEVICE_MAX_CLOCK_FREQUENCY	2445 MHz
CL_DEVICE_PROFILING_TIMER_RESOLUTION	1.000 ns

Portable Computing Language

cpu-haswell-AMD EPYC 7763 64-Core Processor

Examples

Vectorized sum

```
__kernel void vadd(
    __global const float *a,
    __global const float *b,
    __global float *c,
    int verbose) {
  int i = get_global_id(0);
  c[i] = a[i] + b[i];
```

vadd_size = 512

vadd_verbose = 0

```
5 warnings generated.
CL_KERNEL_WORK_GROUP_SIZE                  | 4096
CL_KERNEL_COMPILE_WORK_GROUP_SIZE          | (0, 0, 0)
CL_KERNEL_LOCAL_MEM_SIZE                   | 0 bytes
CL_KERNEL_PRIVATE_MEM_SIZE                 | 0 bytes
CL_KERNEL_PREFERRED_WORK_GROUP_SIZE_MULTIPLE | 8
Send command from host to device   | 1.502 µs
Including data transfer            | 39.118 ms
Execution of kernel                | 29.598 µs
```

Portable Computing Language

cpu-haswell-AMD EPYC 7763 64-Core Processor

```
vadd (generic function with 1 method)
```

Mandelbrot

```
__kernel void mandelbrot(__global float2 *q,
    __global ushort *output, ushort const maxit) {

  int gid = get_global_id(0), it;
  if (gid == 0)
    printf("%d\n", get_num_groups(0));
  float tmp, real = 0, imag = 0;
  output[gid] = 0;
  for(it = 0; it < maxit; it++) {
    tmp = real * real - imag * imag + q[gid].x;
    imag = 2 * real * imag + q[gid].y;
    real = tmp;
    if (real * real + imag * imag > 4.0f)
        output[gid] = it;
```

mandel_size = 512

maxiter = 100

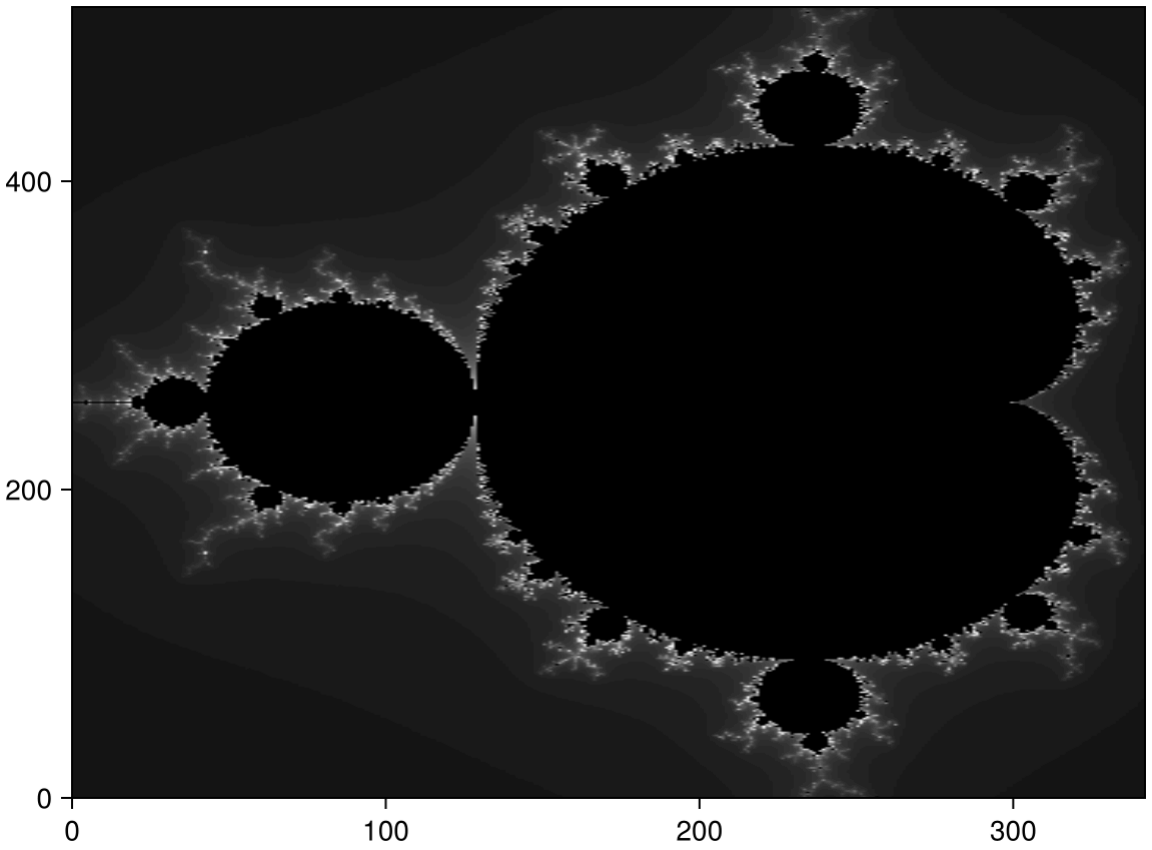
```
q = [ComplexF32(r,i) for i=1:-(2.0/mandel_size):-1, r=-1.5:(3.0/mandel_size):0.5];
```

```
mandel_image = mandel(q, maxiter, mandel_device; global_size=length(q));
```

```
1 warning generated.
CL_KERNEL_WORK_GROUP_SIZE                  | 4096
CL_KERNEL_COMPILE_WORK_GROUP_SIZE          | (0, 0, 0)
CL_KERNEL_LOCAL_MEM_SIZE                   | 0 bytes
CL_KERNEL_PRIVATE_MEM_SIZE                 | 0 bytes
CL_KERNEL_PREFERRED_WORK_GROUP_SIZE_MULTIPLE | 8
57
Send command from host to device  | 1.353 µs
Including data transfer           | 40.420 ms
Execution of kernel               | 12.487 ms
```

Portable Computing Language

cpu-haswell-AMD EPYC 7763 64-Core Processor




```
mandel (generic function with 1 method)
```

```
function mandel(q::Array{ComplexF32}, maxiter::Int64, device; kws...)
    cl.device!(device)
    q = CLArray(q)
    o = CLArray{Cushort}(undef, size(q))

    prg = cl.Program(; source = mandel_source.code) |> cl.build!
    k = cl.Kernel(prg, "mandelbrot")

    timed_clcall(k, Tuple{Ptr{ComplexF32}, Ptr{Cushort}, Cushort},
           q, o, maxiter; kws...)

    return Array(o)
end
```

```
mandel_source = code(Example("OpenCL/mandelbrot/mandel.cl"));
```

Compute π

3.1418869495391846

```
1 mypi()
```

```
CL_KERNEL_WORK_GROUP_SIZE                    | 4096
CL_KERNEL_COMPILE_WORK_GROUP_SIZE            | (0, 0, 0)
CL_KERNEL_LOCAL_MEM_SIZE                     | 0 bytes
CL_KERNEL_PRIVATE_MEM_SIZE                   | 0 bytes
CL_KERNEL_PREFERRED_WORK_GROUP_SIZE_MULTIPLE | 8
Send command from host to device   | 1.393 µs
Including data transfer            | 36.961 ms
Execution of kernel                | 63.051 ms
```

▶ **How to compute π with a kernel ?**

Portable Computing Language

cpu-haswell-AMD EPYC 7763 64-Core Processor

```
mypi (generic function with 1 method)
```

First element

Let's write a simple kernel that returns the first element of a vector in global memory.

```
__kernel void first_el(__global float* glob, __global float* result) {
  int item = get_local_id(0);
  if (item == 0)
    *result = glob[item];
}
```

```
0.59354305f0
```

```
1 first_el(rand(Float32, first_el_len))
```

```
CL_KERNEL_WORK_GROUP_SIZE                  | 4096
CL_KERNEL_COMPILE_WORK_GROUP_SIZE          | (0, 0, 0)
CL_KERNEL_LOCAL_MEM_SIZE                   | 0 bytes
CL_KERNEL_PRIVATE_MEM_SIZE                 | 0 bytes
CL_KERNEL_PREFERRED_WORK_GROUP_SIZE_MULTIPLE | 8
Send command from host to device   | 1.143 µs
Including data transfer            | 28.534 ms
Execution of kernel                | 18.224 µs
```

```
first_el (generic function with 1 method)
```

Portable Computing Language

cpu-haswell-AMD EPYC 7763 64-Core Processor

first_el_len = 16

Copy to local memory

```
__kernel void copy_to_local(__global float* glob, __local float* shared) {
  int global_size = get_global_size(0);
  int local_size = get_local_size(0);
  int item = get_local_id(0);
  shared[item] = 0;
  for (int i = 0; i < global_size; i += local_size) {
    shared[item] += glob[i + item];
  }
}
```

```
1 copy_to_local(copy_global_len, copy_local_len)
```

```
CL_KERNEL_WORK_GROUP_SIZE                  | 4096
CL_KERNEL_COMPILE_WORK_GROUP_SIZE          | (0, 0, 0)
CL_KERNEL_LOCAL_MEM_SIZE                   | 0 bytes
CL_KERNEL_PRIVATE_MEM_SIZE                 | 0 bytes
CL_KERNEL_PREFERRED_WORK_GROUP_SIZE_MULTIPLE | 8
Send command from host to device   | 1.192 µs
Including data transfer            | 30.916 ms
Execution of kernel                | 21.600 µs
```

```
copy_to_local (generic function with 1 method)
```

Portable Computing Language

cpu-haswell-AMD EPYC 7763 64-Core Processor

copy_global_len = 16

copy_local_len = 16

Reduction on GPU

Many operations can be framed in terms of a MapReduce operation.

- Given a vector of data
- It first map each elements through a given function
- It then reduces the results into a single element

The mapping part is easily embarassingly parallel but the reduction is harder to parallelize. Let's see how this reduction step can be achieved using arguably the simplest example of `mapreduce`, the sum (corresponding to an identity map and a reduction with `+`).

Sum

```
(OpenCL = 8.81811, Classical = 8.81811)
```

```
1 local_sum(global_len, local_len, local_code, local_device)
```

```
CL_KERNEL_WORK_GROUP_SIZE                  | 4096
CL_KERNEL_COMPILE_WORK_GROUP_SIZE          | (0, 0, 0)
CL_KERNEL_LOCAL_MEM_SIZE                   | 0 bytes
CL_KERNEL_PRIVATE_MEM_SIZE                 | 0 bytes
CL_KERNEL_PREFERRED_WORK_GROUP_SIZE_MULTIPLE | 8
Send command from host to device   | 1.392 µs
Including data transfer            | 47.402 ms
Execution of kernel                | 18.735 µs
```

▶ **How to compute the sum an array in local memory with a kernel ?**

```
local_sum (generic function with 1 method)
```

Portable Computing Language

cpu-haswell-AMD EPYC 7763 64-Core Processor

global_len = 16

local_len = 16

Blocked sum

(OpenCL = 8.81811, Classical = 8.81811)

```
1 block_local_sum(block_global_len, block_local_len, factor)
```

```
CL_KERNEL_WORK_GROUP_SIZE                  | 4096
CL_KERNEL_COMPILE_WORK_GROUP_SIZE          | (0, 0, 0)
CL_KERNEL_LOCAL_MEM_SIZE                   | 0 bytes
CL_KERNEL_PRIVATE_MEM_SIZE                 | 0 bytes
CL_KERNEL_PREFERRED_WORK_GROUP_SIZE_MULTIPLE | 8
Send command from host to device  | 2.043 µs
Including data transfer           | 41.648 ms
Execution of kernel               | 21.299 µs
```

How to reduce the amount of `barrier` synchronizations ?

Was it beneficial in terms of performance for GPUs like in the case of OpenMP ?

```
block_local_sum (generic function with 1 method)
```

Portable Computing Language

cpu-haswell-AMD EPYC 7763 64-Core Processor

block_global_len = 16

block_local_len = 16

factor = 16

Back to SIMD

- Also called Single Instruction Multiple Threads (SIMT)
- CUDA Warp : width of 32 threads
- AMD wavefront : width of 64 threads
- In general : `CL_KERNEL_PREFERRED_WORK_GROUP_SIZE_MULTIPLE`
- Consecutive `get_local_id()` starting from 0
    - So the thread of local id from 0 to 31 are in the same CUDA warp.
- Threads execute the **same instruction** at the same time so no need for `barrier` .

Warp divergence

Suppose a kernel is executed on a nvidia GPU with `global_size` threads. How much time will it take to execute it ?

```
__kernel void diverge(n)
{
  int item = get_local_id(0);
  if (item < n) {
    do_task_A(); // `a` ns
  } else {
    do_task_B(); // `b` ns
  }
}
```

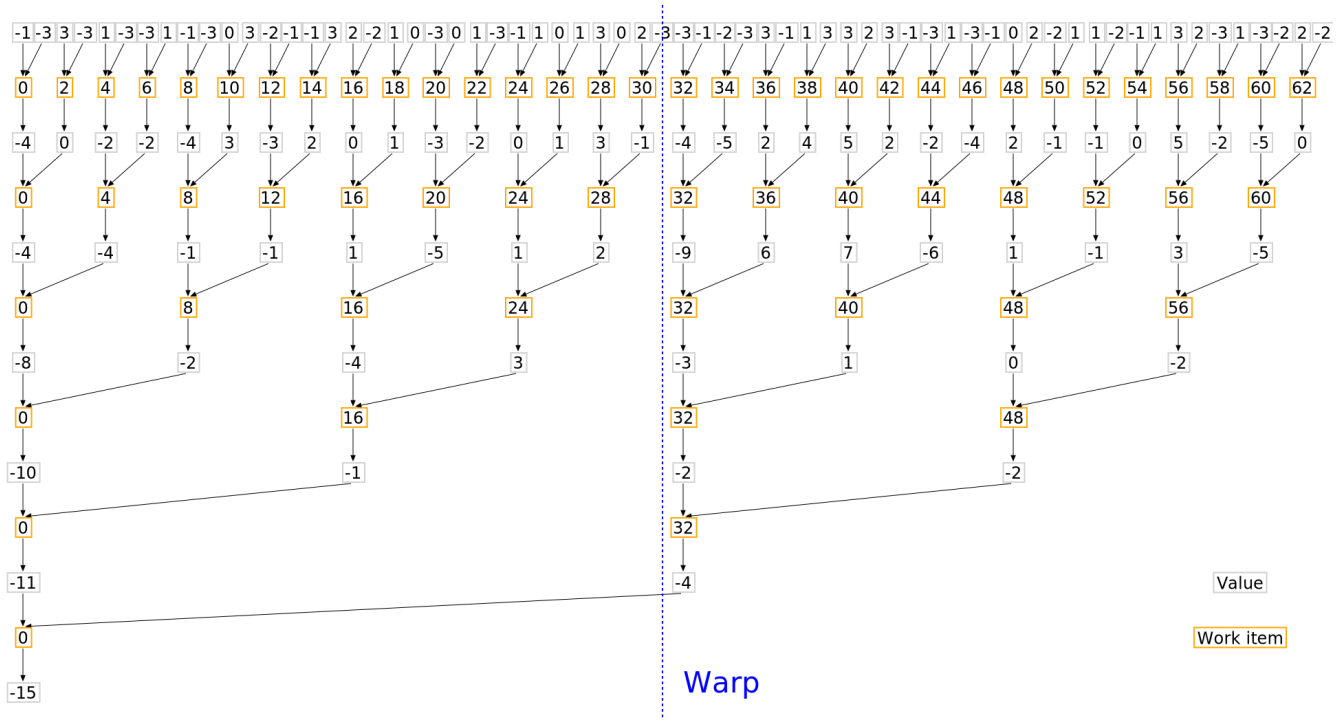
▶ **How much time will it take to execute it if `global_size` is 32 and `n` is 16 ?**

▶ **How much time will it take to execute it if `global_size` is 64 and `n` is 32 ?**

Are the threads that are still active in the same warp for you sum example ?

Warp diversion for our sum

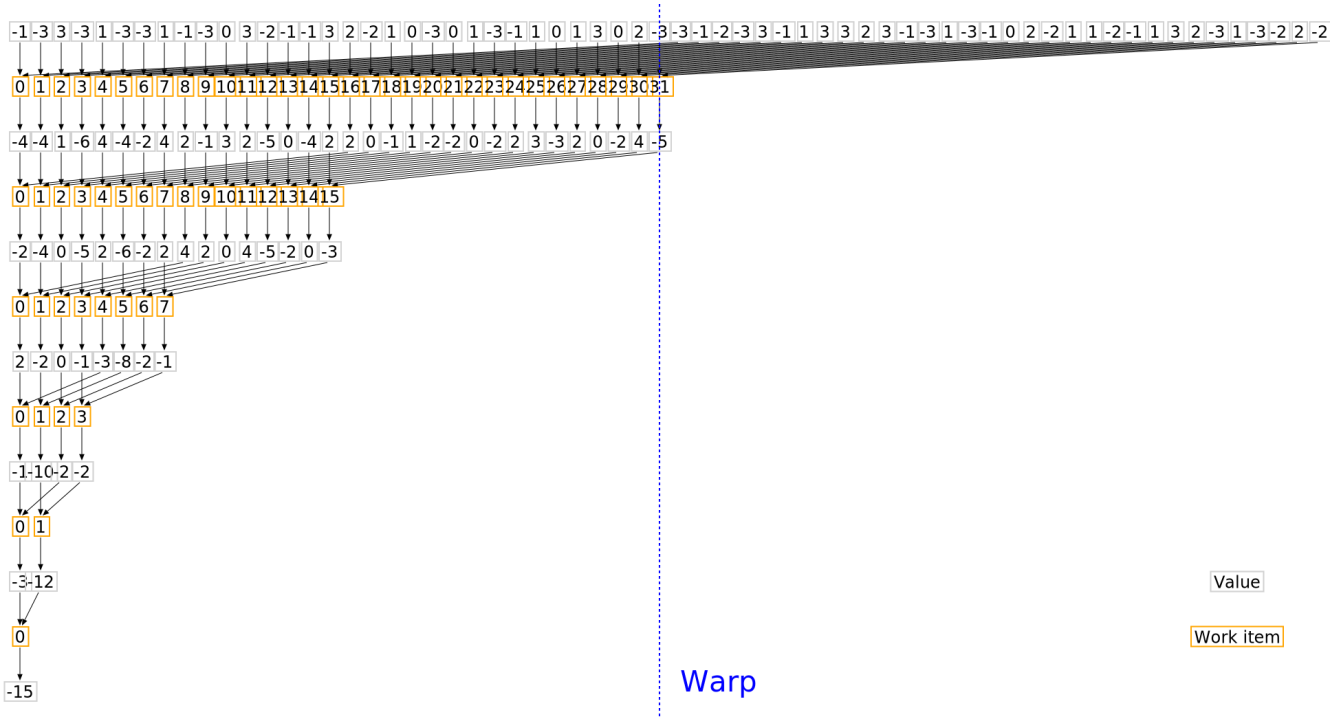
▶ **We are still using different warps until the end. Is that a good thing ?**



How should we change the sum to keep the working threads on the same warp ?

No warp divergence

Now the same warp is used for all threads so we don't need `barrier` and it frees other warps to stay idle (reducing power consumption) or do other tasks.



Reordered local sum

```
__kernel void local_sum(__local float* shared)
{
  int items = get_local_size(0);
  int item = get_local_id(0);
  int stride = items / 2;
  float other_val = 0;
  while (stride > 0) {
    barrier(CLK_LOCAL_MEM_FENCE);
    if (item < stride) {
      other_val = 0;
      if (item + stride < items)
        other_val = shared[item+stride];
      shared[item] += other_val;
    }
    stride /= 2;
  }
}
```

▸ (OpenCL = 8.81811, Classical = 8.81811)

```
CL_KERNEL_WORK_GROUP_SIZE                    | 4096
CL_KERNEL_COMPILE_WORK_GROUP_SIZE            | (0, 0, 0)
CL_KERNEL_LOCAL_MEM_SIZE                     | 0 bytes
CL_KERNEL_PRIVATE_MEM_SIZE                   | 0 bytes
CL_KERNEL_PREFERRED_WORK_GROUP_SIZE_MULTIPLE | 8
Send command from host to device   | 401.000 ns
Including data transfer            | 42.199 ms
Execution of kernel                | 18.966 µs
```

Portable Computing Language

cpu-haswell-AMD EPYC 7763 64-Core Processor

reordered_global_size = 16

reordered_local_size = 16

SIMT sum

```
__kernel void simt_sum(volatile __local float* shared)
{
  int items = get_local_size(0);
  int item = get_local_id(0);
  barrier(CLK_LOCAL_MEM_FENCE);
  while (items > 1) {
    items /= 2;
    shared[item] += shared[item + items];
  }
}
```

(OpenCL = 2.52133, Classical = 8.81811)

```
local_sum(simt_global_size, simt_local_size, simt_code, simt_device)
```

```
CL_KERNEL_WORK_GROUP_SIZE                  | 4096
CL_KERNEL_COMPILE_WORK_GROUP_SIZE          | (0, 0, 0)
CL_KERNEL_LOCAL_MEM_SIZE                   | 0 bytes
CL_KERNEL_PRIVATE_MEM_SIZE                 | 0 bytes
CL_KERNEL_PREFERRED_WORK_GROUP_SIZE_MULTIPLE | 8
Send command from host to device   | 671.000 ns
Including data transfer            | 32.510 ms
Execution of kernel                | 22.722 µs
```

▶ **Why don't we check any condition on `item`, aren't some thread computing data that won't be used ?**

Portable Computing Language

cpu-haswell-AMD EPYC 7763 64-Core Processor

`simt_global_size` = 16

`simt_local_size` = 16

Beware!

POCL does not synchronize, even for `simt_len <= 8`

▶ **Why do we need `volatile` ?**

Unrolled sum

▶ **How to get even faster performance by assuming that `items` is a power of 2 smaller than 512 and that the SIMT width is 32 ?**

▸ (OpenCL = 2.52133, Classical = 8.81811)

```
1 local_sum(unrolled_global_size, unrolled_local_size, unrolled_code, unrolled_device)
```

```
CL_KERNEL_WORK_GROUP_SIZE                  | 4096
CL_KERNEL_COMPILE_WORK_GROUP_SIZE          | (0, 0, 0)
CL_KERNEL_LOCAL_MEM_SIZE                   | 0 bytes
CL_KERNEL_PRIVATE_MEM_SIZE                 | 0 bytes
CL_KERNEL_PREFERRED_WORK_GROUP_SIZE_MULTIPLE | 8
Send command from host to device           | 721.000 ns
Including data transfer                    | 36.020 ms
Execution of kernel                        | 18.705 µs
```

▶ **How to have portable code using unrolling ?**

Portable Computing Language

cpu-haswell-AMD EPYC 7763 64-Core Processor

unrolled_global_size = 16

unrolled_local_size = 16

Utils

_pretty_time (generic function with 1 method)

```
1 _pretty_time(x) = BenchmarkTools.prettytime(minimum(x))
```

```
timed_clcall (generic function with 1 method)
```

```
function timed_clcall(kernel, args...; kws...)
    info = cl.work_group_info(kernel, cl.device())
    # See https://registry.khronos.org/OpenCL/sdk/3.0/docs/man/html/clGetKernelWorkGroupInfo.html
    println("CL_KERNEL_WORK_GROUP_SIZE                    | ", info.size)
    println("CL_KERNEL_COMPILE_WORK_GROUP_SIZE            | ", info.compile_size)
    println("CL_KERNEL_LOCAL_MEM_SIZE                     | ", BenchmarkTools.prettymemory(info.local_mem_size))
    println("CL_KERNEL_PRIVATE_MEM_SIZE                   | ", BenchmarkTools.prettymemory(info.private_mem_size))
    println("CL_KERNEL_PREFERRED_WORK_GROUP_SIZE_MULTIPLE | ", info.prefered_size_multiple)

    # `:profile` sets `CL_QUEUE_PROFILING_ENABLE` to the command queue
    queued_submit = Float64[]
    submit_start = Float64[]
    start_end = Float64[]
    cl.queue!(:profile) do
        for _ in 1:num_runs
            evt = clcall(kernel, args...; kws...)
            wait(evt)

            # See https://registry.khronos.org/OpenCL/sdk/3.0/docs/man/html/clGetEventProfilingInfo.html
            push!(queued_submit, evt.profile_submit - evt.profile_queued)
            push!(submit_start, evt.profile_start - evt.profile_submit)
            push!(start_end, evt.profile_end - evt.profile_start)
        end
    end
    println("Send command from host to device  | $(_pretty_time(queued_submit))")
    println("Including data transfer           | $(_pretty_time(submit_start))")
    println("Execution of kernel               | $(_pretty_time(start_end))")
end
```

num_runs = 1

```
  Activating project at `~/work/LINMA2710/LINMA2710/Lectures`
```

```
1 using OpenCL, pocl_jll # `pocl_jll` provides the POCL OpenCL platform for CPU
  devices
```