Parallel Computing with GPUs

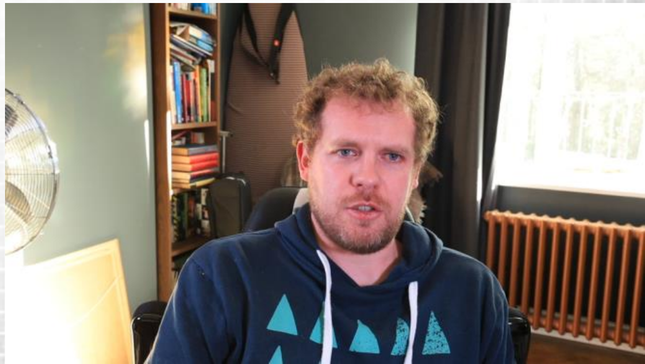
Performance
Part 2 - L1 Cache

The University Of Sheffield.

Dr Paul Richmond
http://paulrichmond.shef.ac.uk/teaching/COM4521/

This Lecture (learning objectives)

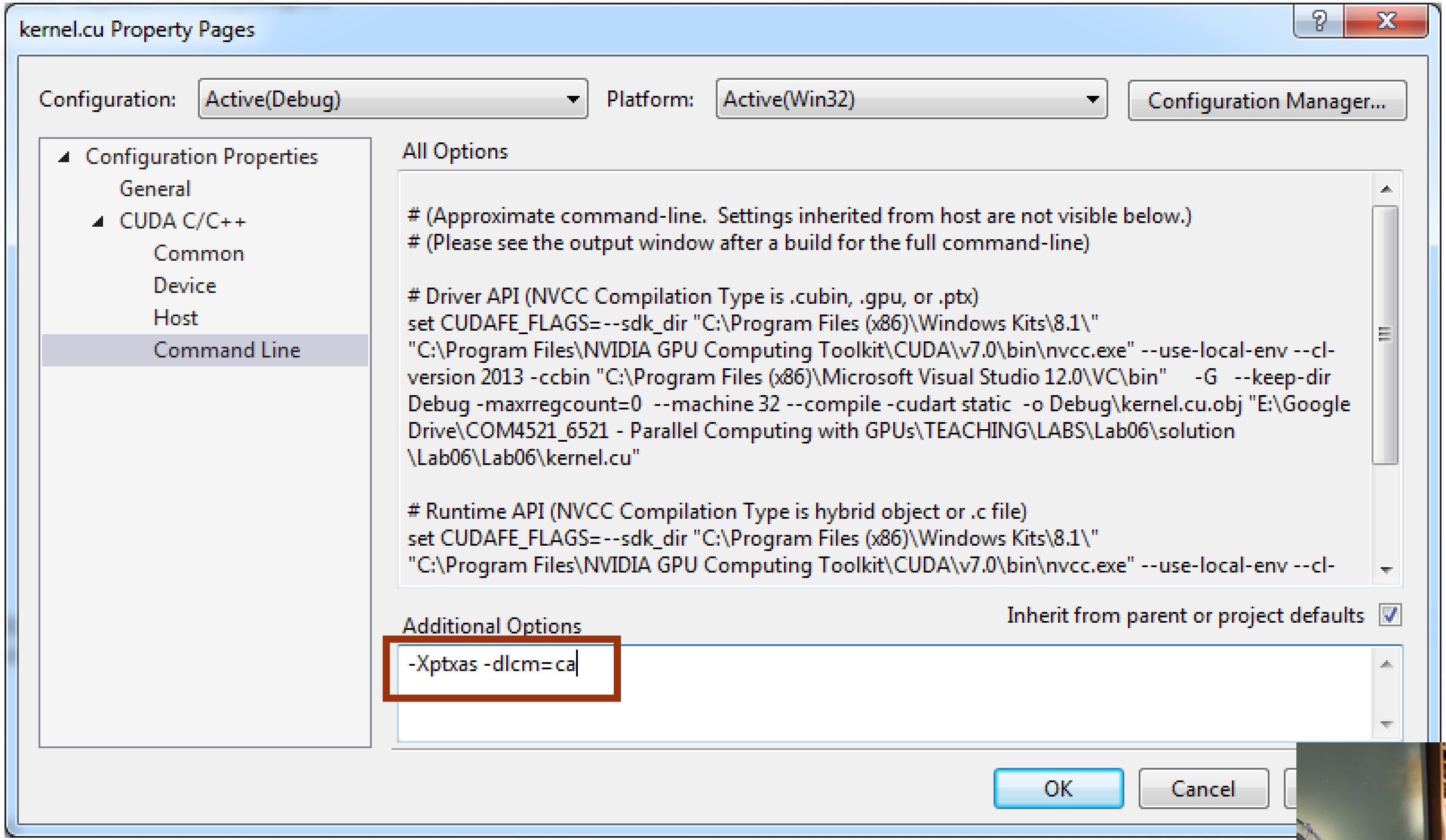
- ❑The L1 Cache
    - ❑Describe how to enable global L1 caching
    - ❑Analyse the performance implications of example access patterns



L1 Cache

- Can utilise L1 via unified L1/Texture/Read-only memory
- Can also be enabled globally
  - What affect does this have on performance of memory movement?
    - Can be good in certain circumstances
      - Coalesced access with adjacent warps reading same data
    - Can also be bad
      - Un-coalesced access performance is worse
      - Increases over-fetch
  - Does my card support global L1 Caching?
    - Check globalL1CacheSupported and localL1CacheSupported CUDA device properties
      - Maxwell 5.2 reports globalL1CacheSupported false when in fact true!
  - Enabling L1 caching of global loads
    - Pass the `–Xptxas -dlcm=ca` flag to nvcc at compile time
      - -dlcm=cg can be used to disable L1 on devices which use it by default

Enabling L1 Cache in Visual Studio



kernel.cu Property Pages

Configuration: Active(Debug)

Platform: Active(Win32)

All Options

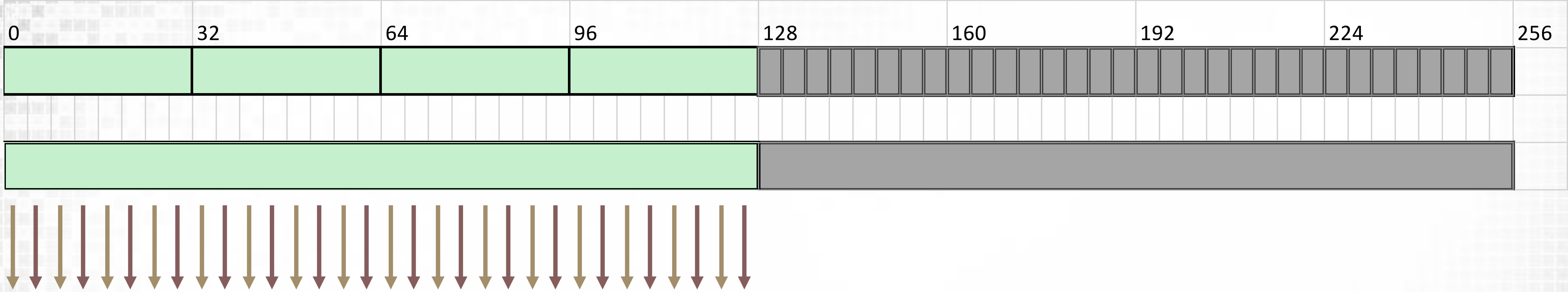
```
# (Approximate command-line.  Settings inherited from host are not visible below.)
# (Please see the output window after a build for the full command-line)

# Driver API (NVCC Compilation Type is .cubin, .gpu, or .ptx)
set CUDAFE_FLAGS=--sdk_dir "C:\Program Files (x86)\Windows Kits\8.1\"
"C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v7.0\bin\nvcc.exe" --use-local-env --cl-
version 2013 -ccbin "C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin"     -G   --keep-dir
Debug -maxrregcount=0  --machine 32 --compile -cudart static  -o Debug\kernel.cu.obj "E:\Google
Drive\COM4521_6521 - Parallel Computing with GPUs\TEACHING\LABS\Lab06\solution
\Lab06\Lab06\kernel.cu"

# Runtime API (NVCC Compilation Type is hybrid object or .c file)
set CUDAFE_FLAGS=--sdk_dir "C:\Program Files (x86)\Windows Kits\8.1\"
"C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v7.0\bin\nvcc.exe" --use-local-env --cl-
```

Additional Options

Inherit from parent or project defaults ☑

```
-Xptxas -dlcm=ca
```

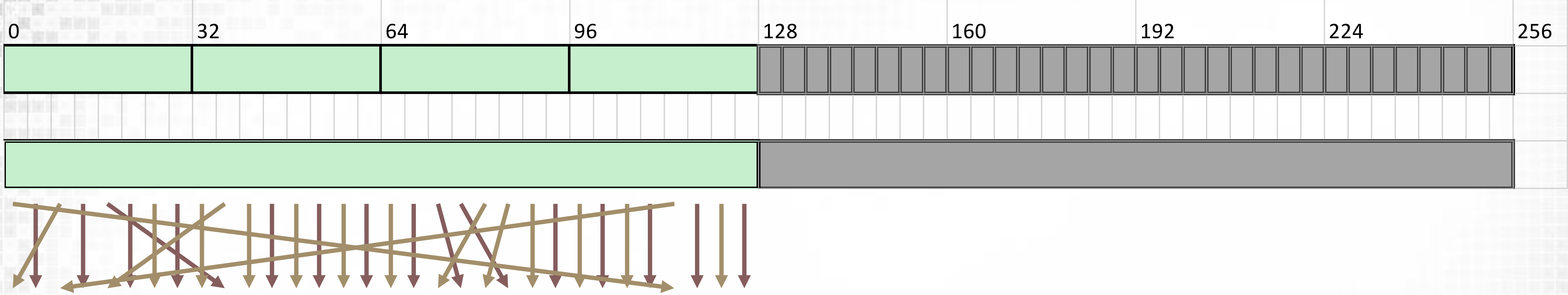
L1 Coalesced Memory Access



```
__global__ void copy(float *odata, float* idata)
    int xid = blockIdx.x * blockDim.x + threadIdx.x;
    odata[xid] = idata[xid];
}
```

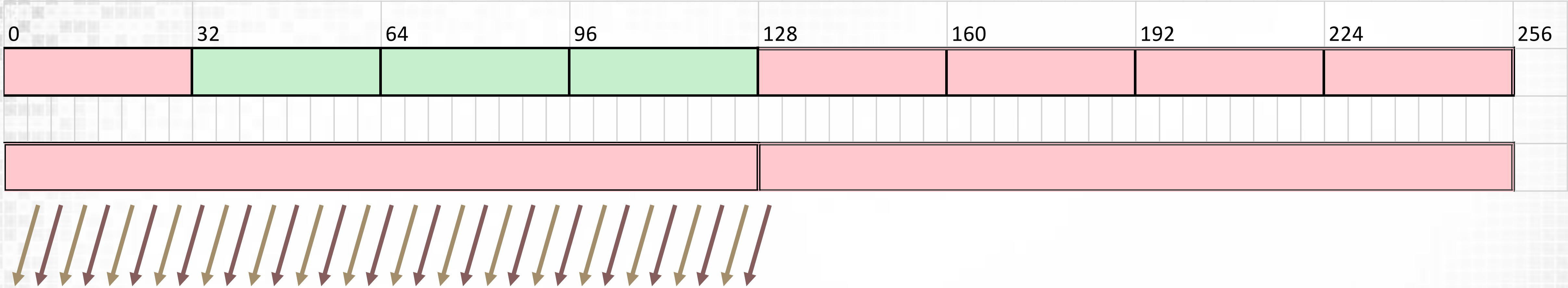
- ❑All addresses fall in one 128B cache line
  - ❑Single transaction
- ❑100% bus utilisation

L1 Permuted Memory Access



- ❑Any thread within the warp can permute access
- ❑Same as L2

L1 Offset Memory Access



```
__global__ void copy(float *odata, float* idata)
    int xid = blockIdx.x * blockDim.x + threadIdx.x + OFFSET;
    odata[xid] = idata[xid];
}
```

- ❑ If memory accesses are offset then parts of the cache line will be unused (shown in red) e.g.
    - ❑ 2 transactions of 256B of which 128B is required: 50% utilisation

- ❑ For strided and random access performance is much worse with L1

Summary

- The L1 Cache
  - Describe how to enable global L1 caching
  - Analyse the performance implications of example access patterns

- Next Lecture: Occupancy