UPPSALA UNIVERSITET

Understanding our code

Carl Nettelblad

2020-06-05

UPPSALA UNIVERSITET

A look back

- We discussed OpenMP Target, Thrust, Cuda, MinPy, afnumpy, Numba
  - If you already know the related technology, go for that:
    - OpenMP => OpenMP Target
    - C++ STL => Thrust
    - numpy => MinPy, afnumpy, CuPy
  - Numba and Cuda are tools of choice if you really want to face the GPU
- `./notebook.py -- gpulibs` brings working numba and cupy, another ”numpy look-alike”
- `./notebook.py -- gpulibs af.sif` brings working afnumpy

UPPSALA UNIVERSITET

CuPy

- Another re-implementation of parts of numpy on GPU
- But with possibility to write inline CUDA C++:

```
>>> kernel = cp.ElementwiseKernel(
...     'float32 x, float32 y',
'float32 z',
...     '''if (x - 2 > y) {
...       z = x * y;
...     } else {
...       z = x + y;
...     }''',
...     'my_kernel’
... )
```

```
>>> l2norm_kernel =
cp.ReductionKernel(
...     'T x',  # input params
...     'T y',  # output params
...     'x * x',  # map
...     'a + b',  # reduce
...     'y = sqrt(a)',  # post-
reduction map
...     '0',  # identity value
...     'l2norm'  # kernel name
... )
```

UPPSALA UNIVERSITET

Using Numba and CuPy

UPPSALA UNIVERSITET

Understanding our code

- For any programming, debugging is important
  - To some extent, the GPU environment is more challenging
- It is good to have tools to verify behavior, but also to ensure correctness
- For performance, measurements and profiling are always useful
  - Even more challenging in a GPU context

UPPSALA UNIVERSITET

The desperate person's debugger

- The GPU is a separate chip, with separate memory
- Unless you have a separate rendering surface, you have no way to send information to the user
- You can not call arbitrary host APIs to write to files etc
- But…
  - Nvidia realized that this was a challenge
  - There *is* a special `printf` function that ”just works”
  - Buffer size is limited
    - Can still be useful to check the value of some specific variable (not of all threads!) or simply if some code is ever run

UPPSALA UNIVERSITET

The real debugger

- The standard gnu debugger gdb has a special version for CUDA code, `cuda-gdb`
  - Can step through both CPU and GPU code
  - gdb is not always the most user-friendly tool
  - There are various wrappers around gdb, such as ddd
  - Many of these can be configured to work against `cuda-gdb`

UPPSALA UNIVERSITET

Unfortunately

- LLVM has a known problem where local variables from OpenMP Target go missing when using `cuda-gdb`

```
(cuda-gdb) b 29
Breakpoint 1 at 0x40212b: file openmptarget.cpp, line 29.
(cuda-gdb) r
Starting program: /home/nettel/sesegpu/openmptarget
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/x86_64-linux-gnu/libthread_db.so.1".
[New Thread 0x2aaab4604700 (LWP 7099)]
Loading file /home/nettel/.keras/datasets/mnist.npz
OpenMP target.
Procesing 10000 * 10000 elements
[New Thread 0x2aaab4805700 (LWP 7105)]
[Switching focus to CUDA kernel 0, grid 3, block (1,0,0), thread (64,0,0), device 0, sm 0,
warp 5, lane 0]
Thread 1 "openmptarget" hit Breakpoint 1,
__omp_offloading_29_c7b6529_main_l19<<<(128,1,1),(128,1,1)>>> ()
    at /home/nettel/sesegpu/openmptarget.cpp:30
30                  for (int y = 0; y < side; y++)
(cuda-gdb)
```

UPPSALA UNIVERSITET

Compiler flags

- `clang++` needs the flag `-g` (or slightly better `-ggdb3`)
- For `nvcc`, specify `-g -G` to include debugging information for host and device
- Optional: Remove `-O3` and replace with `-O0` or `-Og`
  - Reduces optimization. This can make it easier to keep track of data and execution order.
  - `-G` in nvcc also disables optimization. Alternate option `-lineinfo` tries to add debug information to optimized code.

UPPSALA UNIVERSITET

General gdb commands

- `r`(un) start program
- `c`(ontinue) resume running
- `n`(ext) step to next line
- `b`(reak point) *lineno* set breakpoint at this line
- `print` *variable* print the value of a variable (or some other expression)
- `info locals` list all local variables
- `where` print call stack

UPPSALA UNIVERSITET

Cuda-specific `cuda-gdb` commands

- `info cuda` … info on threads for current sm, warps for current sm, lanes in current warp, etc, etc, etc…
- `cuda` … change currently focused thread/sm/lane/…

- Use help command with both of these

UPPSALA UNIVERSITET

info cuda warps

```
  Wp Active Lanes Mask Divergent Lanes Mask Active Physical PC Kernel BlockIdx First Active ThreadIdx
Device 0 SM 2
   0        0xffffffff           0x00000000 0x000000000000cb10      0  (1,0,0)               (0,0,0)
   1        0x00000001           0xfffffffe 0x000000000000c770      0  (1,0,0)               (0,1,0)
   2        0x00000001           0xfffffffe 0x000000000000c990      0  (1,0,0)               (0,2,0)
   3        0xffffffff           0x00000000 0x000000000000cbb0      0  (1,0,0)               (0,3,0)
   4        0xffffffff           0x00000000 0x000000000000cbf0      0  (1,0,0)               (0,4,0)
   5        0xffffffff           0x00000000 0x000000000000cb90      0  (1,0,0)               (0,5,0)
*  6        0xffffffff           0x00000000 0x0000000000000360      0  (1,0,0)               (0,6,0)
   7        0xffffffff           0x00000000 0x000000000000cbf0      0  (1,0,0)               (0,7,0)
   8        0x00000000           0x00000000                n/a    n/a      n/a                 n/a
   9        0x00000000           0x00000000                n/a    n/a      n/a                 n/a
```

- This breakpoint was set some time into running a kernel.
- Observations: 8 warps running on this SM from one block. Maximum of 16 blocks per SM, 32 warps per SM.
- *Occupancy*

UPPSALA UNIVERSITET

Occupancy

- Each execution unit has a limited number of registers.
- Each block defines its maximum usage of registers/warp and and shared memory per block.
  - If a warp function is complex, usage of these resources might increase, decreasing occupancy.
- `nvcc` has a flag `-resource-usage` reporting these numbers.
  - Numbers were high.
- Try again using the nvcc flag `-lineinfo`
  - Optimization kicks in.

UPPSALA UNIVERSITET

Another attempt

```
(cuda-gdb) info cuda warps
  Wp Active Lanes Mask Divergent Lanes Mask Active Physical PC Kernel BlockIdx First Active ThreadIdx
Device 0 SM 0
*  0        0xffffffff           0x00000000 0x0000000000000520      0  (0,0,0)              (0,0,0)
   1        0xffffffff           0x00000000 0x0000000000000520      0  (0,0,0)              (0,1,0)
   2        0xffffffff           0x00000000 0x0000000000000520      0  (0,0,0)              (0,2,0)
   3        0xffffffff           0x00000000 0x0000000000000520      0  (0,0,0)              (0,3,0)
   4        0xffffffff           0x00000000 0x0000000000000520      0  (0,0,0)              (0,4,0)
   5        0xffffffff           0x00000000 0x0000000000000520      0  (0,0,0)              (0,5,0)
   6        0xffffffff           0x00000000 0x0000000000000520      0  (0,0,0)              (0,6,0)
   7        0xffffffff           0x00000000 0x0000000000000520      0  (0,0,0)              (0,7,0)
   8        0xffffffff           0x00000000 0x0000000000000470      0  (1,0,0)              (0,3,0)
   9        0xffffffff           0x00000000 0x00000000000004e0      0  (1,0,0)              (0,0,0)
  10        0xffffffff           0x00000000 0x0000000000000480      0  (1,0,0)              (0,1,0)
  11        0xffffffff           0x00000000 0x00000000000004c0      0  (1,0,0)              (0,2,0)
  12        0xffffffff           0x00000000 0x00000000000004c0      0  (1,0,0)              (0,7,0)
  13        0xffffffff           0x00000000 0x0000000000000520      0  (1,0,0)              (0,4,0)
  14        0xffffffff           0x00000000 0x00000000000004c0      0  (1,0,0)              (0,5,0)
  15        0xffffffff           0x00000000 0x00000000000004c0      0  (1,0,0)              (0,6,0)
  16        0x0000000f           0xfffffff0 0x00000000000002a0      0 (79,0,0)              (0,2,0)
  17        0xffffffff           0x00000000 0x0000000000000330      0 (79,0,0)              (0,3,0)
  18        0xffffffff           0x00000000 0x0000000000000310      0 (79,0,0)              (0,0,0)
  19        0xffffffff           0x00000000 0x00000000000002f0      0 (79,0,0)              (0,1,0)
  20        0xffffffff           0x00000000 0x0000000000000390      0 (79,0,0)              (0,6,0)
  21        0xffffffff           0x00000000 0x0000000000000310      0 (79,0,0)              (0,7,0)
  22        0x0000000f           0xfffffff0 0x0000000000000320      0 (79,0,0)              (0,4,0)
  23        0x0000000f           0xfffffff0 0x0000000000000330      0 (79,0,0)              (0,5,0)
  24        0xffffffff           0x00000000 0x0000000000000260      0 (81,0,0)              (0,1,0)
  25        0xffffffff           0x00000000 0x00000000000002f0      0 (81,0,0)              (0,2,0)
  26        0xffffffff           0x00000000 0x0000000000000240      0 (81,0,0)              (0,3,0)
  27        0xffffffff           0x00000000 0x0000000000000280      0 (81,0,0)              (0,0,0)
  28        0xffffffff           0x00000000 0x0000000000000240      0 (81,0,0)              (0,5,0)
  29        0xffffffff           0x00000000 0x00000000000002f0      0 (81,0,0)              (0,6,0)
  30        0xffffffff           0x00000000 0x00000000000002b0      0 (81,0,0)              (0,7,0)
  31        0xffffffff           0x00000000 0x0000000000000340      0 (81,0,0)              (0,4,0)
```

UPPSALA UNIVERSITET

cuda-memcheck

- With highly parallel code, with manually computed indices, it's very easy to write to the wrong memory addresses
- `cuda-memcheck` is a tool that was initially developed to address this
  - Runs the code in its original form
  - But with lots of extra checks (far slower than normal)
    - 6 minutes rather than 400 ms for our the prelab2 `./cuda` program
  - Provides report on invalid accesses
- Since this tool is working directly on the GPU, it can be used with any binary running on the GPU
  - E.g. Python with numba, TensorFlow, if you believe that they misbehave

UPPSALA UNIVERSITET

Tool modes

- cuda-memcheck has several subtools
  `cuda-memcheck --tool x`
  - `memcheck` is the default, checking invalid writes.
  - `initcheck` verifies that memory is properly initialized before the first read.
  - `synccheck` verifies that synchronization primitives do not have hidden usage errors (such as not all lanes participating).
  - `racecheck` checks for race conditions in shared memory, e.g. where writes and reads on different threads do not have a barrier.

Nice extra flags

```
--leak-check <full|no> [Default : no]
        Print leak information for CUDA allocations.
        NOTE: Program must end with cudaDeviceReset() for this to work.

--track-unused-memory <yes|no> [Default : no]
        Check for unused memory allocations. This requires initcheck tool.
```

UPPSALA UNIVERSITET

Performance

- Now we ended up analyzing performance/allocation behavior using the debugger
- The Nvidia Nsight profiler can do this and more.
  - Nsight System
    - Interaction between CPU and GPU, multiple kernels
  - Nsight Compute
    - Focus on individual kernels

UPPSALA UNIVERSITET

GPU Performance Counters

- During the last years, side-channel attacks have become more important for hardware vendors
  - “Is there something you can measure that will allow private data to leak between processes?”
  - If you can get very exact data on cache or instruction processing behavior, that can form a side-channel
  - Something to be aware of when you write code
    - Is my code ever used in a security-sensitive context? Can timing/energy usage/other factors for processing some input tell a bit too much about the internals?
  - Spectre and Meltdown are well-known Intel CPU cases
  - Long story short: Sadly, by default you need to be an admin user to measure these things nowadays.

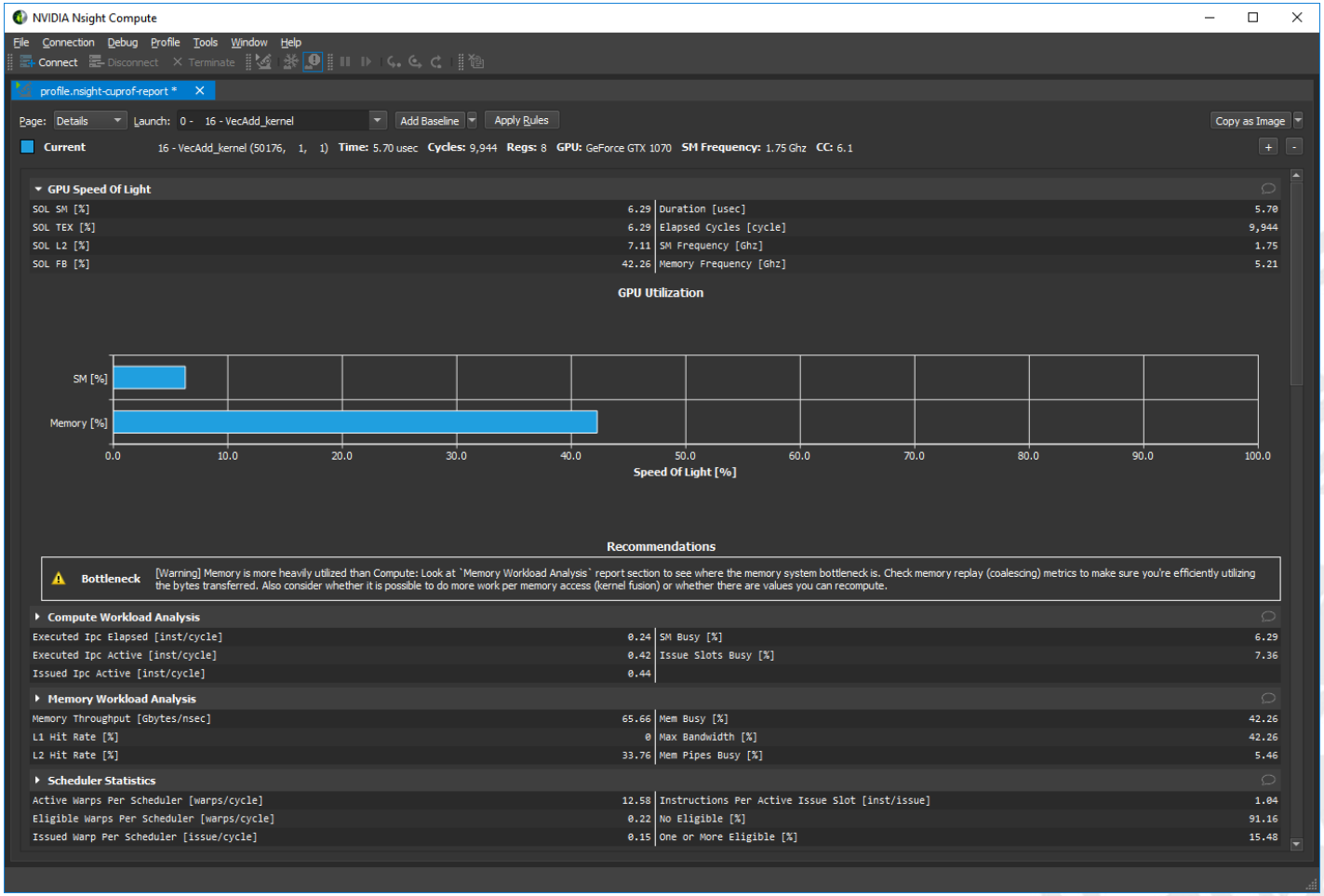
UPPSALA UNIVERSITET

What can Nsight Compute do?

- Measure performance
- Total time usage
- Total number of memory fetches
  - Cache hit rates
  - From different memory levels
- What instructions are stalling?
- Occupancy
  - Threads per SM
- Usage of execution resources
  - IPC, overall, when active
  - What units are used
    - Our Cuda kernels would show that the floating-point units are not untilized at all

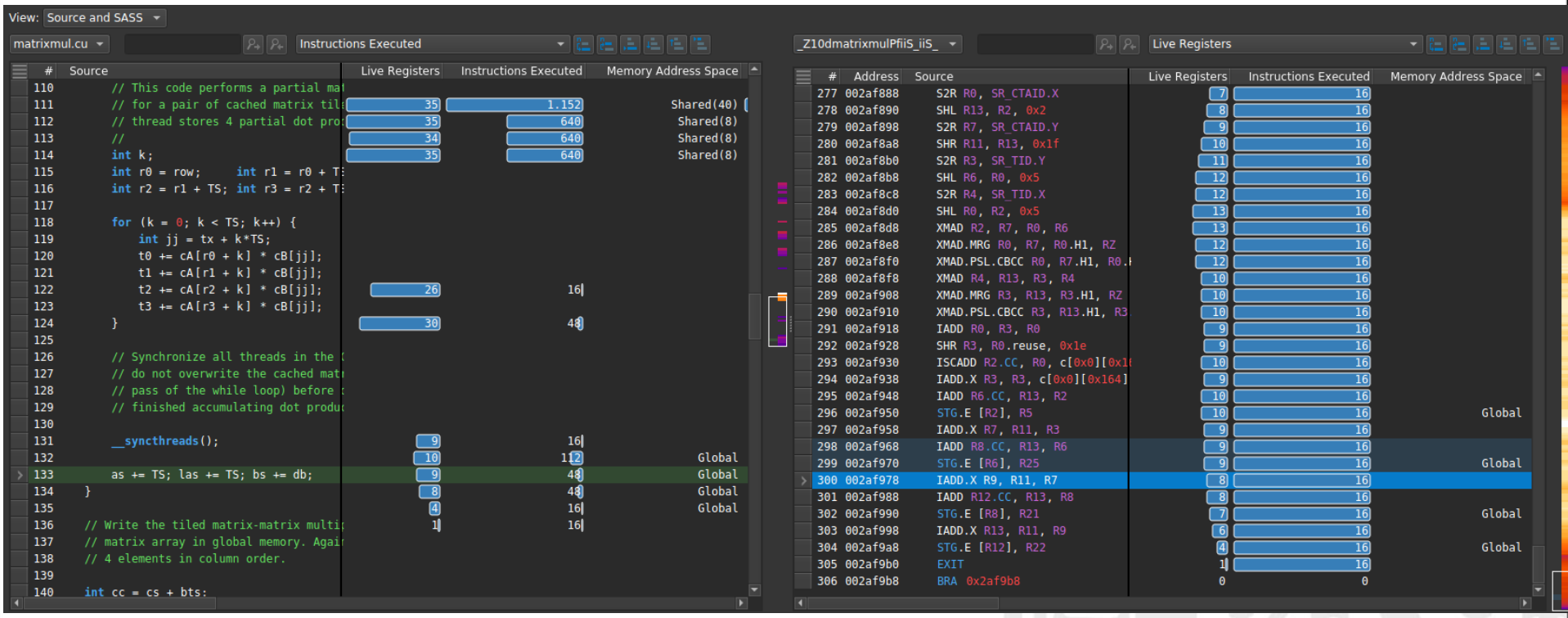
UPPSALA UNIVERSITET

Overall resource usage



UPPSALA UNIVERSITET

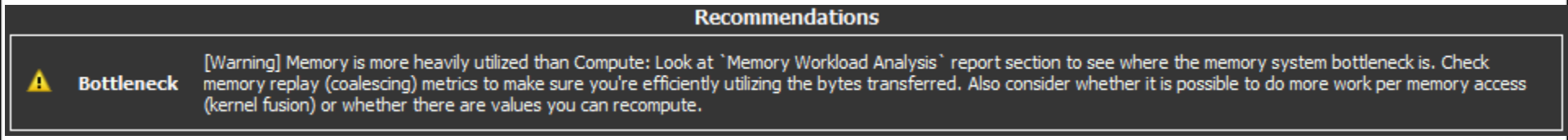
Profile per code line



- Time, register pressure and memory accesses neatly in one view, with assembler and source

UPPSALA UNIVERSITET

Actionable advice



- A set of default rules that are applied. Replacing basic human intuition until one gets a feeling for it.

UPPSALA UNIVERSITET

Conduct experiments

- Try to be systematic in what you try if you try to improve performance on GPU.
  - What am I expecting to happen?
  - Why am I expecting it to happen?
  - How can I prove/disprove my hypothesis?
  - How can I isolate one aspect of behavior from all interacting factors, while still giving valid insights for the “real-world” case?
  - Is the scaling as expected if datasets increase in size?
  - If I claim that performance is optimal, is it at all close to maximum specs?
    - Hint: For our examples, it’s not.
- The profiler makes it easier to formulate and evaluate an experiment.
  - And refine your hypothesis for the next attempt.

UPPSALA UNIVERSITET

Report your experiments

- If you come from an applied field, making the point that a new approach is faster and/or more flexible and/or more accurate can seem like a nuisance
- In the project, that's the core.
  - What application problem did you solve?
  - What computational changes did you make?
  - How did you ensure that those changes were correct?
  - What quantitative results can you present?
  - What alternate solutions might exist? ("Future work" or just options that you considered)

UPPSALA UNIVERSITET

Project

- So, ideally:
  - A problem from your own research.
    - Or, preferably, at least vaguely related to it.
  - An implementation scaling onto GPU or scaling out into cloud resources.
  - Reporting on your experiments and conclusions.
  - At least 3 pages, at most 12, IEEE Transactions template.
    - We prefer you keep it simple. It has to be legible, but not overedited.
  - One week of work. Deadline August 31.
    - Feedback and advise might be slower during July and August…

UPPSALA UNIVERSITET

Now

- Think about what you want to do. Maybe go back to some of the lab content and previous slides for inspiration.
- Ideally, can you do a proof of concept right now to check whether an idea is feasible?
- Reach out to me or Salman at any time between now and 15.15.

UPPSALA UNIVERSITET

At 15.15

- Write down at least a sentence on what you intend to do.
  - Put it in a slide.
  - Share it with us on screen at 15.15.
  - We'll have a joint discussion on possible pitfalls, or even if there are synergies between multiple suggested projects.
  - Reports and code should be clearly attributable to single individuals, though.
    - I.e. if two of you develop some piece of code, you each have to use it/test it/extend it in a different way.

- You're free to change your project topic later, but we want you to leave the course week with an initial idea on what to do.

UPPSALA UNIVERSITET

Thank you!

- Questions?