# OpenCL Practical 3 – optimisation

This practical will give you a chance to practice optimising kernels to make them run more quickly. The main objectives are to learn about:

- how to use the different levels of OpenCL's memory hierarchy
- how to use multiple kernels at once

1. Log in to the head node, e.g. ssh username@gpu.hector.ac.uk
2. Change to the prac3 directory: “cd ~/opencl_course/prac3”
3. Look at the Makefile to see how it works then type “make”
4. Submit jobs to the GPUs via the queue manager using ‘qsub’, e.g. “qsub jobSub3”
5. Keep track of where your jobs are in the queue with “qstat”
6. Have a look at the output that’s produced in jobSub3.oxxx – how fast were the CPU and simple GPU results?

7. Add a new kernel corresponding to the “one row of C per work-item” on page 23-24 of lecture 3. Don’t forget to make the necessary changes to the host code. What is its performance?
8. Add a new kernel corresponding to the “private row of A per work-item” on page 27 of lecture 3. How does its performance compare to the previous examples?
9. Add a new kernel corresponding to the “private row of A per work-item, local columns of B per work-group” on page 30-31 of lecture 3. How does its performance compare to the previous examples?

# COMPETITION!

**What's the fastest you can make yours go? (Without cheating!)**

===== **Sequential**, matrix mult (dot prod), order 1000 on host CPU ======
9.34 seconds at **214.1** mflops

===== **OpenCL**, matrix mult, **C(i,j) per work item**, order 1000 ======
1.08 seconds at **1,851.1** mflops

===== **OpenCL**, matrix mult, **C row per work item**, order 1000 ======
1.72 seconds at **1,162.6** mflops

===== **OpenCL**, matrix mult, **C row, A row in priv mem**, order 1000 ======
0.50 seconds at **3,964.9** mflops

===== **OpenCL**, mat mult, **C row, priv A, B cols local**, order 1000 ======
0.52 seconds at **3,829.6** mflops

Results from one Nvidia Fermi C2050