**GPU, Many-core, and Cluster Computing**

**Moore’s law**Moore's law refers to an observation made by Intel co-founder Gordon Moore in 1965. He noticed that the number of transistors per square inch on integrated circuits had increased at a rate of roughly a factor of two per year since their invention. It is a statement on complexity, not speed.

Processor makers delivered increases in clock rates and instruction-level parallelism, so that single-threaded code executed faster on newer processors with no modification. However as transistors get smaller and smaller they get restricted by physical limitations, meaning Moore’s law will reach a point where it no longer holds. In order to manage this, processor makers favour multi-core chip designs, and software has to be written in a multi-threaded manner to take full advantage of the hardware.

**HPC**HPC (high performance computing) concerns how to make code run faster. There are three classic objectives of HPC:

* Throughput and efficiency - Reduce cost because results are available faster. Means scientists and engineers can do more experiments in a given time frame or study more parameters
* Response time – Enables better computation for urgent situations such as tsunami prediction or in medical applications
* Problem size – allows new insight through never before seen resolution/zoom into scales. Enables scientists to solve problems even bigger/harder.

To achieve these objectives requires an increase of concurrency and diversity on all hardware levels. Since individual chips are not becoming faster, HPC challenges enter the mainstream and therefor so do GPUs and multi/manycores.

**Inside a core**The central compute facility (compute registers) can run one operation on multiple variables in one step. This is known as vectorisation.

The instruction stream interpretation can logically run more instructions per cycle than supported by the compute units. This is done via:

* Pipelining - allows successive steps of an instruction sequence to be executed in turn by a sequence of modules able to operate concurrently, so that another instruction can be begun before the previous one is finished.
* Branch prediction - tries to guess which way a branch (e.g. an if-then-else structure) will go before this is known for sure

All three of the above techniques can be found in GPUs in a slightly modified manner.

Cores are not directly connected to the memory, but through a L1/L2 cache. Done via cache-aware algorithms.

**Inside a processor**Possible configurations for an 8 core processor:

* Shared memory multicore machine - All eight cores share a common L3 cache (cf. future sessions) as well as one memory controller/memory channel (one joint access to RAM)
* Shared memory with non-uniform data access – two sockets with 8 cores. This processor then connects to memory and to external hardware. Hamilton (Durham’s supercomputer) has this configuration. The configuration makes it a GPGPU (general purpose GPU i.e. not just for graphics).

Similar design decisions to the second configuration are repeated for a GPU