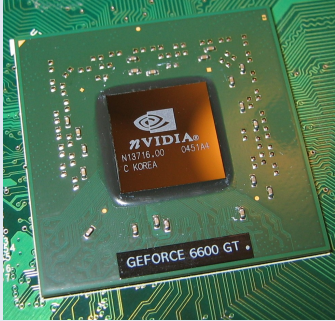


Graphics Processing Units

Graphics Processing Units, originally developed for rendering output for a monitor, have progressed to enable calculations of AI and Physics.



GPUs are especially good for

- manipulating large amounts of data in the same way (SPMD);
- real-time high definition video decoding;
- mapping light rays in 3D scenes, and much more.

With the right techniques we can apply GPUs to large and complex statistical problems.

Optimising

The GPU has evolved to become more readily available as a general purpose processor, with the help of nVidia and the CUDA toolkit, we can convert parts of standard C programs into CUDA kernels which will run on the GPU.

The design of the GPU is such that a small amount of memory is allocated to groups of threads called *blocks*. These blocks grouped into a *warp*. Threads in a warp will be executed simultaneously with the same instructions.

Limited memory availability leads to complications. In particular, should we use fat threads (many data, few threads, less memory transfers) or thin threads (small data, many threads, maximum parallelism, data transfer required during simulation).

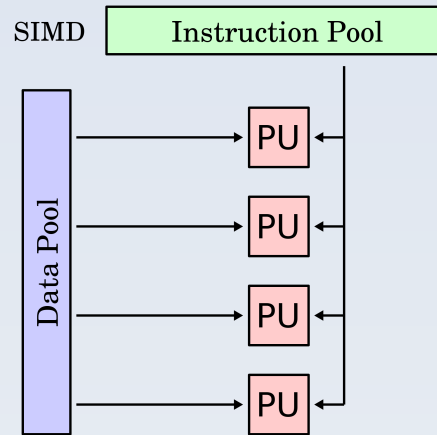
References

- [1] NVIDIA CUDA Education (<http://goo.gl/gSH8k>)
- [2] Holmes, C., Lee, A. GPU Stochastic Simulation for statistical data analysis (<http://goo.gl/vYIXR>)
- [3] Berkut NVIDIA GPU Photograph (<http://goo.gl/4U8ND>)
- [4] Burnet, C. SIMD Diagram (<http://goo.gl/vuSvV>)

Parallel Computing

GPUs are optimised to perform the same action on a large number of vectors at once. For example, simulating the path of individual bricks or particles from a simulated explosion where each brick reacts in a similar way but with differing initial conditions.

In terms of parallel computing this is called Single Instruction, Multiple Data, or SIMD.



Standard parallel computing often focuses on running separate applications, or parts of, within a separate core, processor, or machine and combining the various outputs of each.

GPUs change this in that data is treated as a large object and is manipulated as a whole. This is obviously case dependant.

Results & Conclusions

This graph shows a basic GPU and a basic CPU performing a simple Monte Carlo simulation. The CPU performs at the same rate no matter how much data is given, but the GPU will work faster when given more data up to the card and simulation's individual limiting factors.

With the right techniques many simulations and experiments can be solved more quickly by using a GPU, in some situations speed ups of 100x have been realised. Furthermore, the cost of running a GPU is *much* lower than that of running a CPU cluster. These factors make it worth while developing new techniques to run simulations on a GPU rather than a CPU.

Monte Carlo Experiments

Monte Carlo(MC) experiments are a class of algorithms that rely on repeated random sampling.

A simple example is Monte Carlo Integration. Here we randomly select uniform values over a fixed area and accept values based on some criteria based on some criteria

- Generate 2 uniform random numbers
- Calculate your acceptance number
- Compare this value to the second number
- Is the result of the equation greater than the second random number?
- If yes, add one to the successful hits counter
- Iterate.

This gives an estimate of the integral as the value of (Successful Hits/Total Hits) within the region of the random numbers you generated.

