GPU Computing

WHY USE GPUS

Fundamentally power dissipation requirements are limiting single-core CPU speeds but we selfishly want yet faster computing.

Graphics card Processing Units (GPUs) in recent years have moved from being highly specialised for graphics processing to being high speed general computing systems.

This process was started by NVIDIA but has been followed by AMD and Intel. Of particular advantage is that Graphics cards have a large number (100s) of individual processors compared to the standard CPU (with 4 to 12 cores). With GPUs having a similar speed clock to modern CPUs this in theory leads to processing speed ups when looking at similar cost/power-consumption.

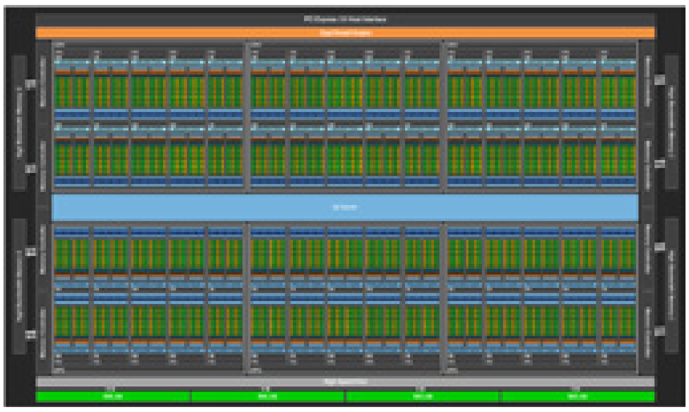

Figure 1: A block diagram of NVidia's Pascal GPU: Each small green square is a FP32 processing core. Each small yellow square is a FP64 core.

In the 1980s the world's most powerful supercomputers were dominated by machines such as the Cray-2 which was a 1.9 GFlops 4-core system; the emphasis was on getting a few cores to run as fast as possible. Today all the world's most powerful supercomputers are massively parallel systems with hundreds-of-thousands or even millions of cores. Some use bespoke cores but some just use a lot of GPUS. For example the current second fastest computer in the world, Titan, a Cray XK7, running at 17.59 PFlops is based on 261,632 NVidia K20x cores.

ENTER CUDA

CUDA is a parallel computing platform and application programming interface (API) model created by NVIDIA. It allows software developers to use a CUDA-enabled graphics processing unit (GPU) for general purpose processing an approach known as GPGPU[1]. The CUDA platform is a software layer that gives direct access to the GPU's virtual instruction set and parallel computational elements.

The CUDA platform is designed to work with programming languages such as C, C++ and Fortran. In form it looks very similar to C. This accessibility makes it easier for specialists in parallel programming to utilize GPU resources, as opposed to previous API solutions like Direct3D and OpenGL, which required advanced skills in graphics programming. Also, CUDA supports programming frameworks such as OpenACC and OpenCL. When it was first introduced by NVIDIA, the name CUDA was an acronym for Compute Unified Device Architecture, but NVIDIA subsequently dropped the use of the acronym.

The remaining disadvantage is that in order to take advantage of the GPU paradigm you need the ability to process tasks with different threads running lockstep in parallel rather than sequentially. As expected this produces limitations both in algorithm design and implementation. Fortunately many common computing tasks can be redesigned so they can be processed in parallel.

1 General-purpose computing on Graphics Processing Units

Processor	GFlops (FP32/FP64)	Cost/$ (Jan '16)	Power/W	Architecture
Xeon E5-2699 v3	700+/700+	4500	145	18 core 2.3GHz
Xeon E5-2698 v3	600/600	3750	135	16 core 2.3GHz
i7-5960x	176/ 176	1000	140	8 core 3GHz
GeForce GTX 950	2308 / 49	200	120	768 cores 1.2GHz
GeForce GTX1080Ti	10608 / 332	800 (Mar '17)	250	3584 cores 1.5GHz
Tesla K40	5000 / 1500	2500	235	2880 cores @ 0.8GHz
Tesla P100	10000 /5000	10000 (Mar '16)	250	3584 cores @ 1.4GHz
Jetson TX2	1500	400 (Mar '17)	10	256 cores GPU + Quad core Arm A57

Table 1: Performance of various processors (Grey = CPUs, Blue = PC based GPU, Green = Embedded SoC)

THEORETICAL PERFORMANCE

As will be seen, the theoretical 32-bit (single precision) processing capability of GPUS now significantly exceeds those of even high cost CPUs, even when dealing with relatively cheap GeForce gaming orientated GPUs.

With 64-bit (double precision) the performance of the cheap GeForce GPUs falls to around that of the CPUs but NVidia provide Tesla class GPUs which keep the majority of their 32-bit performance when performing 64-bit calculations. This is all whilst maintaining similar power consumption to the CPU.

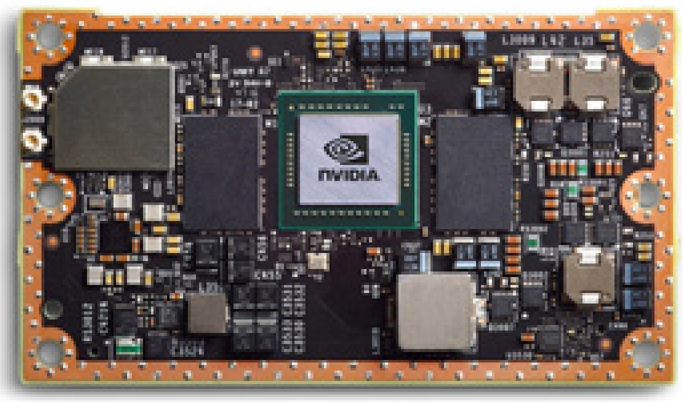

Figure 2: NVidia Jetson TX2 System-on-chip

PRACTICAL GPU PERFORMANCE

Whilst GPUs have a massive theoretical performance advantage over CPUs this is only for those problems which are truly parallel computations. In practice most problems have some sequential elements which restrict the speed-up available from a GPU.

Remember however that for the CPUs too the theoretical performance is only possible if all the cores can be used simultaneously. Whilst this is easier to achieve than true lockstep parallel processing it still requires significant work. Where this does not happen the performance of a single CPU core remains stubbornly stuck at under 40GFlops, over 100 times slower than many GPUs.

Secondly many real world problems are often limited by how fast data can be transferred to/ from the GPU. Tasks which require a lot of data transfers are harder to speed up than those which are dominated by the sheer amount of computing required.

Debunking the 100x GPS vs. GPU Myth: An Evaluation of Throughput Computing on CPU and GPU (Lee, Kim et al, ISCA'10 June 19-23, 2010, Saint Malo, France) showed that the type of application made a big difference as to the degree of speed up possible on GPUs vs CPUs. For the chips examined (i7 and GTX 280) it found that GPUS were typically 3 times faster than CPUs across numerous different types of computing problems with a range between 1:0.5 (GPU slower than a CPU) through to 1:14.9 depending on the precise problem.

EMBEDDED GPUS

The final line in Table 1 above shows the alternative way of imagining GPUs. Instead of looking at increasing the maximum amount of processing power look at the problem as being reducing the total amount of electrical power required to perform a particular computing task such as machine vision or running the processing for an airborne radar.

There are many situations where low-SWaP (low Size, Weight and Power) is essential. Traditionally this meant either low computing power (a single embedded micro-processor or DSP) or a bespoke FPGA solution with the associated lack of flexibility but with a complicated long and high cost development.

Embedded GPUs such as the NVidia Jetson TX2 offers another possibility. The platform is credit card sized, 87 x 50mm, weighs 88g and takes a maximum of 15W. Via its GPU this low-SWAP solution still offers processing power of the order of a mains-powered high-end PC but with a much quicker and cheaper development path than an FPGA.

SUMMARY

Some problems are very efficient on GPUs. Some are more efficient on modern CPUs. Raw GFLOP processing power seems to drive the average case.

In general for single-precision mathematics if the problem can run in parallel even cheap gaming orientated GPUs are frequently faster than CPUs. For double-precision mathematics NVidia's gaming orientated GPUs have deliberately weakened performance, though still similar to a high class CPU. Their Tesla class GPUs however have similar performance processing double-precision as for single-precision mathematics. SoCs such as NVidia's Jetson TX2 offers the possibility of Intel i7 level processing but whilst only requiring low power.

Author: Peter Debenham
Date: June 2017