GPU vs CPU Smackdown : The Rise of Throughput-Oriented Architectures

Friday, December 3, 2010 at 9:20AM

Todd Hoff in Paper, gpu, hpc

In some ways the original Amazon cloud, the one most of us still live in, was like that really cool house that when you stepped inside and saw the old green shag carpet in the living room, you knew the house hadn't been updated in a while. The network is a little slow, the processors are a bit dated, and virtualization made the house just feel smaller. It has been difficult to run high bandwidth or low latency workloads in the cloud. Bottlenecks everywhere. Not a big deal for most applications, but for many high performance applications (HPC) it was a killer.

In a typical house you might just do a remodel. Upgrade a few rooms. Swap out builder quality appliances with gleaming stainless steel monsters. But Amazon has a big lot, instead of remodeling they simply keep adding on entire new wings, kind of like the Winchester Mystery House of computing.

The first new wing added was a CPU based HPC system featuring blazingly fast Nehalem chips, virtualization replaced by a close to metal Hardware Virtual Machine (HVM) architecture, and the network is a monster 10 gigabits with the ability to specify placement groups to carve out a low-latency, high bandwidth cluster. Bottlenecks removed. Most people still probably don't even know this part of the house exists.

The newest addition is a beauty, it's a graphics processing unit (GPU) cluster as described by Werner Vogels in Expanding the Cloud - Adding the Incredible Power of the Amazon EC2 Cluster GPU Instances . It's completely modern and contemporary. The shag carpet is out. In are Nvidia M2050 GPU based clusters which make short work of applications in the sciences, finance, oil & gas, movie studios and graphics.

To get a feeling of the speed involved read BillMcColl's comment:

> *Cloudscale is now able to ANALYZE a SINGLE STREAM of entity events at a rate of TWO MILLION EVENTS PER SECOND (150MB/sec) on an 8-node 10-GigE Amazon cloud cluster. That's well over ONE TRILLION EVENTS per week.*

Having both CPU and GPU clusters seems a bit strange. Why have two? Dr. Vogels does a good job explaining the reasoning:

- GPUs work best on problem sets that are ideally solved using massive fine-grained parallelism, using for example at least 5,000 - 10,000 threads. To be able build applications that exploit this level of parallelism one needs to enter a very specific mindset of kernels, kernel functions, threads-blocks, grids of threads-blocks, mapping to hierarchical memory, etc. Configuring kernel execution is not a trivial exercise and requires GPU device specific knowledge. There are a number of techniques that every programmer has grown up with, such as branching, that are not available, or should be avoided on GPUs if one wants to truly exploit its power.

> Modern CPUs strongly favor lower latency of operations with clock cycles in the nanoseconds and we have built general purpose software architectures that can exploit these low latencies very well. Now that our ability to generate higher and higher clock rates has stalled and CPU architectural improvements have shifted focus towards multiple cores, we see that it is becoming harder to efficiently use these computer systems. One trade-off area where our general purpose CPUs were not performing well was that of massive fine grain parallelism. Graphics processing is one such area with huge computational requirements, but where each of the tasks is relatively small and often a set of operations are performed on data in the form of a pipeline. **The throughput of this pipeline is more important than the latency of the individual operations**. Because of its focus on latency, the generic CPU yielded rather inefficient system for graphics processing. This lead to the birth of the Graphics Processing Unit (GPU) which was focused on providing a very fine grained parallel model, with processing organized in multiple stages, where the data would flow through. The model of a GPU is that of task parallelism describing the different stages in the pipeline, as well as data parallelism within each stage, resulting in a highly efficient, high throughput computation architecture.

The ACM has a timely article about using GPUs for high performance computing ACM: Understanding Throughput-Oriented Architecture by Michael Garland and David Kirk:

> *Scalability is the programmer's central concern in designing efficient algorithms for throughput-oriented machines. Today's architectural trends clearly favor increasing parallelism, and effective algorithmic techniques must scale with hardware parallelism. Some*

> *techniques suitable for four parallel threads may be entirely unsuitable for 4,000 parallel threads. Running thousands of threads at a time, GPUs are a powerful platform for exploring scalable algorithms and a leading indicator for algorithm design on future throughput-oriented architectures.* ***GPUs are specifically designed to execute literally billions of small user-written programs per second***.

What matters here are two things: tools and jobs.

First, there's another exotic tool in the toolbox to solve difficult problems in ways very different than what we are used to. This along side the original recipe cloud, MapReduce, and the CPU Cloud, offers enormous flexibility when architecting systems.

Second, for a surprisingly large number of problems there is now a ready supply of GPU supercomputeryness. With supply there can be demand and not that many people know how to program GPUs. Programming GPUs is a specialized skill. GPUs are a very different kind of device. It's nothing like using your typical threading library, eventing infrastructure, and message passing library. Now that GPU processors are so readily available we'll need GPU programmers to make use of all that power. Something to think about as a potential career direction.

In retrospect the title of this article seems to imply it's either CPU or GPU and that there will be one winner. That wasn't the intent. The core idea here is to say here's something many people aren't familiar with, it's now easily and publicly available, here are some more sources to learn from, and maybe you can use these new capabilities to your advantage in your project. But that's kind of boring :-) Clearly GPUs aren't general purpose processors. The context here is more what your average developer now has easy access to in the cloud, not what they can do with a lot of expertise and special equipment in their own server room.

Related Articles

- A Couple More Nails in the Coffin of the Private Compute Cluster
- HPC in the Cloud with GPGPUs by James Hamilton
- Designing Efficient Sorting Algorithms for Manycore GPUs
