An Atomic Tesla: Avoiding the Power Wall

R. G. Edgar,[1,2] **M. D. Miller,**[2] **R. E. Parrott,**[2] **S. Sahrakorpi,**[2] **J. Sircar**[2]

[1]Initiative in Innovative Computing, Harvard University,
[2]School of Engineering and Applied Sciences, Harvard University

rge21@seas.harvard.edu

http://scigpu.org/

Tackling the 'Power Wall'

Modern CPUs have a voracious appetite for electical power, creating the 'power wall' problem for supercomputer design. This is discussed in great detail in DARPA's Exascale Computing study (Kogge et al., 2008). Modern GPUs offer FLOP-per-Watt performance far better than contemporary CPUs, and NVIDIA's CUDA toolkit has enabled researchers to show how useful GPUs can be for general purpose computation. GPUs merit serious consideration as the basis of an Exascale machine.

However, GPUs must be supported by a conventional CPU-based host, to run an operating system and manage the GPU device. Most contemporary clusters use high-end server processors for this task. However, if one can adopt the 'Reverse Acceleration' model developed at Los Alamos National Laboratory for Roadrunner (Pakin, Lang and Kerbyson, 2010), these CPUs are likely to be spending most time idling, leading us to consider using a low power processor to manage the GPU.

Hardware

We compared the performance of an Intel Atom 330 CPU to conventional clusters containing Intel Xeon L5410 ('Harpertown') and Intel Xeon E5540 ('Gainestown') CPUs. The CPUs were connected to rackmounted GPUs, Tesla S1070s for the Atom and Harpertown clusters and QuadroPlex S4s for the 'Gainestown' cluster.

Unfortunately, we were not able to procure equipment with identical PCIe interfaces. The Atom system had a PCIe 1.0 x16 slot, the Harpertown system a PCIe 2.0 x16 slot and the Gainestown system a PCIe 2.0 x8 slot.

Benchmarking

The three key metrics for our study are:

1. The latency of memory transfers between CPU and GPU
2. The bandwidth of these transfers
3. The latency of kernel launches on the GPU

Transfers between CPU and GPU

We copied contiguous blocks of data between pinned memory on the host and the GPU device in our bandwidth test. Twenty such copies were combined in a single timing, and each timing was performed twenty times for each data size. We show the results for host-to-device copies in Figure 1 (device-to-host copies are similar).

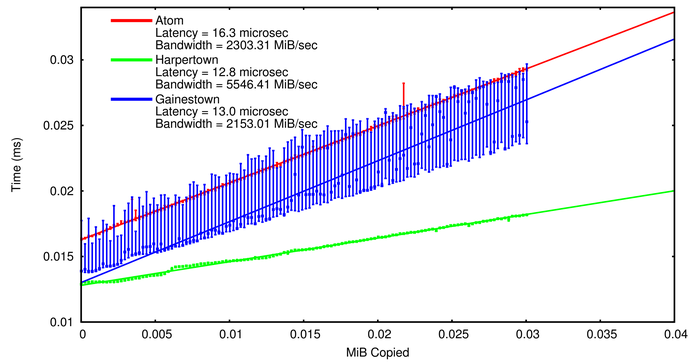

Figure 1: *Host to Device Bandwidth Timings for Atom, Harpertown and Gainestown-based systems. Each data point is based on repeating twenty copies twenty times. The error bars show the median and quartile timings. The Atom system had a PCIe 1.0 x16 slot, the Harpertown system a PCIe 2.0 x16 slot, and the Gainestown system a PCIe 2.0 x8 slot. Latency and bandwidth values are from fitting a line to the median times.*

The Gainestown results are much noisier than the other two, probably due to this shared system being under heavier load than the other two. However, we see that the Harpertown system (with its PCIe 2.0 x16 bus) was approximately twice as fast as the other two, as expected from the nominal bus speeds. Furthermore, the latency of transfers for the Atom system is slightly greater than the other two systems. We experimented with the BIOS settings of a Harpertown system, nominally changing the PCIe 2.0 bus to PCIe 1.0. This cut the bandwidth to that of the Atom system, while retaining the Harpertown latency.

Kernel Launch Latency

To measure the kernel launch latency, we used a kernel which zeroed out a device array. By adjusting the number of threads and thread blocks launched, the kernel launch latency could be determined. Figure 2 shows our results for the three systems we tested. We see that the Atom system had a kernel launch latency of around $9\ \mu\mathrm{s}$, while the Harpertown and Gainestown systems both had latencies around $4.5\ \mu\mathrm{s}$.

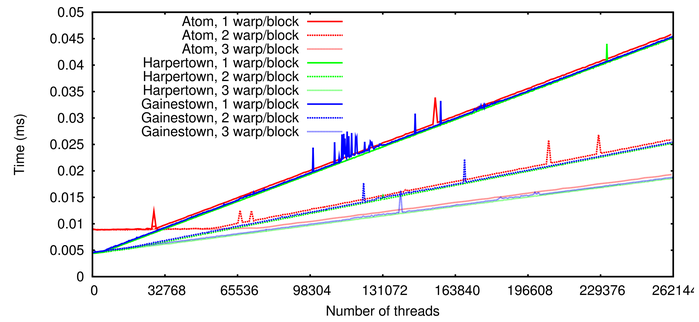

Figure 2: *Kernel Timings for Atom, Harpertown and Gainestown-based systems. A simple kernel zeroed out an array on the device, with varying numbers of threads and thread blocks launched. The Atom system had a PCIe 1.0 x16 slot, the Harpertown system a PCIe 2.0 x16 slot, and the Gainestown system a PCIe 2.0 x8 slot. The kernel launch latency is inferred to be the y-intercept of the curves, around $9\ \mu\mathrm{s}$ for the Atom system and around $4.5\ \mu\mathrm{s}$ for the Harpertown and Gainestown systems.*

Conclusions

The Atom-based system held its own in these tests, with little impact on the operation of the GPU. Power draw for the entire Atom system (including its integrated GeForce 9400M chipset and hard drive) was less than $50\ \mathrm{W}$ under load. Further work is needed using production codes and with identical PCIe interfaces. However, for codes with serial sections small enough to make Amdahl's Law irrelevant, Atom-based host systems are viable candidates.

Acknowledgements

The authors acknowledge financial support provided through NSF Award PHY-0835713. Harvard University has been named a CUDA Center of Excellence by NVIDIA, and has received several hardware donations which were used in this work. Some results were obtained using the longhorn machine of the Texas Advanced Computing Center, through allocation TG-AST100022.