Leveraging system architecture to optimize Deep Learning (DL) and Artificial Intelligence (AI)-based processing of massive data sets.

[1]Christopher M. Sullivan, Michaela E. Buchanan[1], Mazen N. Alotaibi[1], Kenneth B. Lett[1], Brett M. Tyler[1]

[1]Center for Genome Research and Biocomputing, Oregon State University, Corvallis, OR

As data scientists move forward integrating high throughput data collection and monitoring devices into the life and environmental sciences, differences are becoming apparent across available CPU and GPU architectures that are transforming analysis of massive data sets. These architectures are creating new opportunities beyond the narrow range of platforms that have been available over the past two decades. These new technologies are becoming ever more important as current infrastructures are challenged by applications of machine learning (ML) and deep learning (DL) algorithms to massive data sets. These increasingly large data sets, from numerous application domains, challenge computing technologies to scale to avoid limits to data analysis. High performance computing groups that support a diverse set of research projects are at the forefront of addressing these challenges. Many groups working with ML or DL approaches are using Graphics Processing Units (GPU) based technologies to help overcome processing limits. In most cases, these GPU technologies are based on add-on cards to current hardware infrastructure. Here at the Center for Genome Research and Biocomputing (CGRB) at Oregon State University, we have found that these expansion cards have limits when interacting with large amounts of data or need to interact with the CPU and memory bus. The CGRB and its collaborators in the life and environmental sciences have established multiple ML- and DL-based algorithms and have identified several examples in which GPU performance was

heavily limited by system bus and architecture. In this paper we describe evaluation of several GPU-based architectures for processing of massive audio, video and DNA sequence data sets. We describe how optimizing CPU-GPU integration enables efficient processing of massive data sets on a scale needed for improved analysis results.

The first algorithm we assessed was a tool we created at the CGRB to quantify owl species in forests, based on sound recordings of owl calls. This work was conducted as part of a project to monitor Northern Spotted Owl populations that has been funded by the US Forest Service for many years (https://www.fs.fed.us/r6/reo/monitoring/nso). The Northern Spotted Owl (NSO) Monitoring Program conducts long-term monitoring of NSO populations and habitat to determine the effectiveness of federal forest management on maintaining and restoring habitat conditions necessary to support viable populations of NSO on federally-administered lands throughout its range. The project, a collaboration with OSU professor Katie Dugger and US Forest Service researchers, Damon Lesmeister and Zack Ruff, began with sound data collected using over 1500 autonomous recording units spread out across the forest. Approximately 250TB of data were collected every season. We segmented these data into images and processed them through a Convolutional Neural Network (CNN) that can currently identify 11 species of owls as well as a few species of mammals. Initially, we used NVIDIA-based GPU technologies and found them to work the best for many of these types of image processing applications, compared to CPU-based applications. For example, when we used CPUs with tensorflow, it took over 50 hours for training a test run compared to 2 hours on a PCIe-located Nvidia Tesla K80. At that time, we were using well established PCIe-based cards including the Tesla K80 and K40 GPU's in a standard Supermicro server that supported eight

GPU cards. The sound data were segmented on the CPU, then segments containing bird calls were sent to the GPU for identification using the CNN. While the PCIe cards worked well, we quickly realized that the quantity of data we were sending to the GPUs was overwhelming the bandwidth of the PCIe bus. This was a serious limitation as we needed to scale up the number of recorders and collection sites around the forest. Therefore, we explored GPU-based technologies that were installed directly on the system board rather than on the PCIe bus. This led us to Power8 servers with NVlink buses created by IBM and NVIDIA. The NVlink bus massively increased GPU to GPU throughput was as well as accelerated communication between the GPUs and system components like the CPU and memory. Because the new IBM hardware utilized little endian byte code and we were using other cross platform software stacks like CUDA and Tensorflow, we were able to run the exact same code on these new systems without issues. Once we moved onto NVLink machines with Tesla P100 GPU's, we immediately observed a 4-5x performance increase compared to the PCIe Tesla K80 cards we were using on the x86 platforms (Table 1).

Run Type	K80 on x86	P100 on IBM OpenPOWER
Training the CNN	30:06:13	06:18:07
Generating predictions	00:45:35	00:08:44
Full data processing	01:22:00	00:39:50

Table 1: Table of run times for PCIe based GPU hardware using x86 server's vs NVLink on IBM OpenPOWER. Server configuration for x86 was a standard Supermicro server with 4x K80 GPUs and 2x 1TB SSD drives for scratch space. The OpenPOWER configuration was a ppc64le server with 2x P100 GPUs and 2x 1TB SSD drives for scratch space.

In order to verify that the performance increase was not simply due to the difference in GPUs being used, we compared the performance with a PCIe-based P100 installed in standard Dell R740 servers and found there was still a 3x performance increase. Next, we tested a new Power9 server with an NVLink V100 GPU and compared this new IBM system to a V100 GPU card on an x86 system (configurations found in Table 2, below) and observed a 2x speed increase on the NVLink system.

After comparing GPU architectures for CNN performance, we compared CPU performance for the segmentation task. To do this we ported our software to the POWER9 system to take advantage of IBM Power machines that provide up to 160 threads per server. We aimed to compare the IBM systems to the x86 based machines we had previously used to host the GPU hardware. To create the fairest comparison, we obtained a machine that would match the IBM Power machine in terms of CPU and memory speed as well as support for a large number of GPUs; this was a Supermicro 4U server with a Xeon (Broadwell 2.2 GHz) CPU, 1 TB RAM and 2x1 TB SSD. (Table 2). Comparison of these machines allowed us to evaluate the systems for pre-processing of the data, particularly segmentation, prior to the processing steps on the GPUs (Table 3).

At this point we learned that NVIDIA was working on a new system called the DGX that would support a flavor of NVLink on x86 platforms that would interconnect GPUs, though not the CPU. To more formally compare the performance of this technology with the IBM Power and SuperMicro machines, we obtained a DGX2 machine (Table 2) and ran a series of standard benchmarking tests on the three machines (Table 3). The comparison revealed a 2-4x performance increase on the IBM Power machine compared to the Supermicro. The CPU

speeds on the DGX2 were faster and the system had a larger number of usable NVMe devices than the AC922. Nonetheless, the DGX was about 3x slower on CPU tasks and 1.5x slower on the input-output bus tasks than the IBM Power machine. On the other hand, the DGX2 was faster on tasks using system memory (Table 3).

IBM AC922 server with dual POWER9 rev_2.1 @ 2.250 GHz including 1TB DDR4 with 4x NVIDIA V100-32GB GPGPU (NVLink Bus) on the PPC64LE architecture
Supermicro 4U server with dual Intel® Xeon® CPU E5-2650 v4 @ 2.20GHz including 512G DDR4 with 4x NVIDIA V100-32GB GPGPU (PCIe BUS)
NVIDIA DGX2 server with dual Intel® Xeon® Platinum 8168 2.70GHz including 1.5TB DDR4 with 16x NVIDIA V100-32GB GPGPUs (NVLink Bus) on the x84_64 architectures.

Table 2: Machines used in the final timed tests. Several other server configurations were discussed throughout this document to show the evolution of working with different GPU to server configurations. These other configurations were not used in the final time test and were not included in this table.

Machine	Run Type	Architecture	Test Size	Throughput (total events)
AC922	CPU	Power9 (2.25GHz)	4-Thread	121389
Supermicro 4U Server	CPU	Xeon (Broadwell 2.2 GHz)	4-Thread	34994
DGX2	CPU	Xeon (Platinum 8168 2.70GHz)	4-Thread	46254
Machine	**Run Type**	**Architecture**	**Test Size**	**Throughput (total events)**
AC922	MEMORY	Power9 (2.25GHz)	102400MiB	35398559
Supermicro 4U Server	MEMORY	Xeon (Broadwell 2.2 GHz)	102400MiB	41610663
DGX2	MEMORY	Xeon (Platinum 8168 2.70GHz)	102400MiB	53126648
Machine	**Run Type**	**Architecture**	**Test Size**	**Throughput (total events)**
AC922	Input/Output SSD	Power9 (2.25GHz) with SSD (used as single disk)	128 files, 16MiB each 2GiB total file size	118500
Supermicro 4U Server	Input/Output SSD	Xeon (Broadwell 2.2 GHz) with SSD (used as single disk)	128 files, 16MiB each 2GiB total file size	47545
Machine	**Run Type**	**Architecture**	**Test Size**	**Throughput (total events)**
AC922	Input/Output NVMe	Power9 (2.25GHz) with 2x NVMe RAID0 CAPI Bus	128 files, 16MiB each 2GiB total file size	2267206
DGX2	Input/Output NVMe	Xeon (Platinum 8168 2.70GHz) with 8x NVMe RAID0	128 files, 16MiB each 2GiB total file size	1529698

Table 3: Benchmarks showing the ability of different server class hardware that support GPU technology and how they perform. The benchmark runs as many iterations it can on each system within a set amount of time. Looking at the area inside the red box the larger the number in the events the more throughput. There are two I/O tests done to separate the effects of SSD and NVMe on the different servers.

We next evaluated the AC922 and the Supermicro server using a CNN-based tool that requires extensive transactions between the CPU and the GPUs. This tool is used to identify and quantify plankton in the ocean using video data to help understand ocean health. Our collaborators at the Hatfield Marine Science Center at Oregon State University, led by Professor Robert Cowen, uses research ships to drag video cameras through the ocean, collecting video image data tagged with geographical locations. Each research cruise of one week brings back over 80TB of video data. As with the owl data, the video data is segmented into individual images that are then processed by a CNN on GPUs to identify the plankton species in each image. Segmentation produces thousands of small compressed files with 100 -

10,000 or more images in each of them, each tagged with the geographical location. Segmentation of 80 TB of video data can result in 120-160 TB of image files. These image files are sent to the GPU in batches and since they use small amounts of GPU memory, many files can be sent to the GPU at one time. Thus, processing of these files results in numerous transactions between the CPU and GPUs over short periods of time. When we benchmarked this transactional processing of data across the different architectures, we observed a much stronger dependence on bus speeds than tools that just stream data through to the GPU. We observed a 4-5x performance increase when processing this type of data through the NVLink bus of the AC922 compared to the Supermicro machine with the PCIe bus. A speed increase was also seen on the CPU side, as with pre-processing of the owl data, and that increased the overall throughput of this algorithm on the AC922 to 6-8x compared to the Supermicro server. The coupling of segmentation and classification on a single machine with almost 8x performance increase moved the total processing time below the amount of time needed for the ship to travel back to port. In future, this will allow processing of the data on the ship, significantly accelerating progress of the project.

The third GPU-based tool we evaluated was one that can fit onto a GPU and its memory. This evaluation was intended to separate the effects of GPU performance and bus performance. This tool was called CUDA Accelerated Scalable Sequence Aligner (CASSA) and was created to take large quantities of genomic DNA sequence data and align them to a known genome. This task is called High Throughput Sequencing (HTS) data analysis and is used widely to understand how our planet and its inhabitants live or interact, including many aspects of

human health. The data sets used for this test were publicly available human genome sequences obtained from the NIH GenBank repository. For the initial tests, these data sets were limited to 2GB in size to ensure they could fully fit on the NVIDIA V100 32G GPUs, so that there should be no interaction with the bus speed other than first loading the data onto the GPU card. Once the data are loaded, the process is solely limited by the GPU speed. We observed that if the data completely fit onto a single GPU, there was no difference in running the jobs on either architecture; both completed in around 00:01:14. When we increased the size of the data set to 50GB files, well past the 32 GB that would fit onto the memory of the GPU card, we observed a 2-3x performance increase compared to the PCIe based GPU cards (see Figure 1 below). When the AC922 was compared to the DGX2, we observed a small increase of around 1.2x when using the data set that would fully fit onto the GPU, but much of that was starting the process and moving data to the GPU. Once we started to scale up and leverage more than a single GPU for the processing the NVLink produced a significant drop in time on the IBM Power machines compared to the DGX2 (see Table 4).

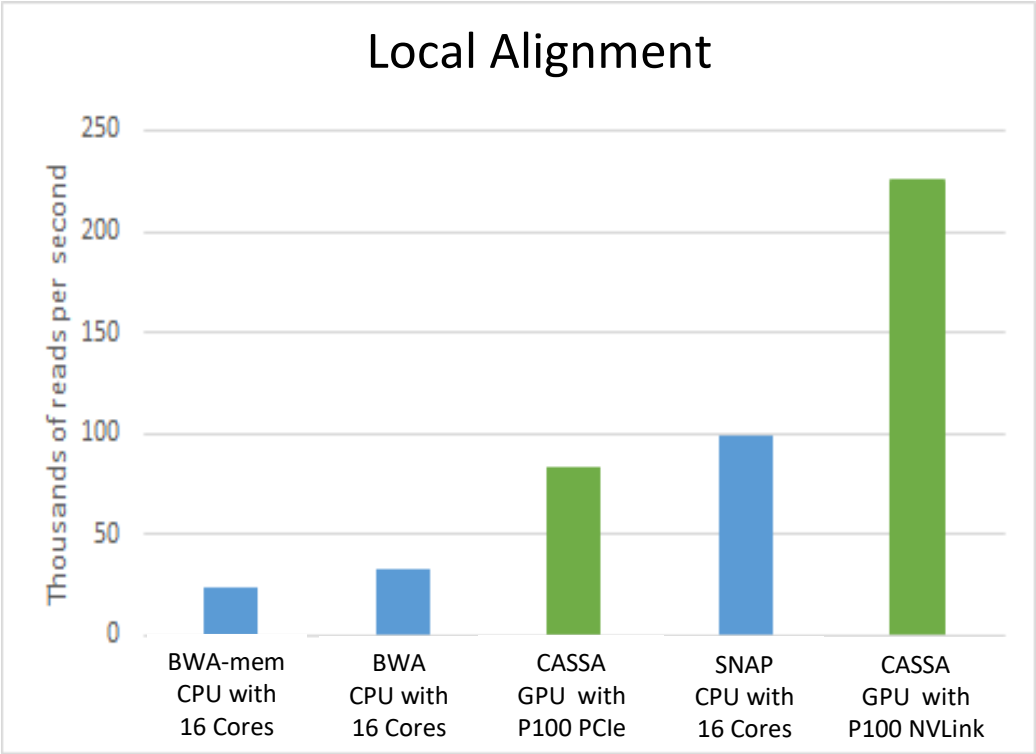

Figure 1: Number of sequence reads aligned to the genome over time compared across algorithms and platforms. The data was raw High Throughput Sequencing (HTS) data and was a single input file of 2G (standard file size) and was aligned to the Human Genome. Each column is a single tool run with a specific hardware configuration. BWA and SNAP are standard HTS alignment tools that leverage the Burrows-Wheeler Algorithm used throughout the research community and use system CPU and memory to process data. CASSA uses GPU technology to do HTS alignment while still using the Burrows-Wheeler Algorithm.

Overall, across all algorithms, when we compared these same Power9 and x86 systems with multiple GPUs on a single server we saw the most dramatic performance improvements (Figure 1; Table 4). As we increased the number of GPUs, we were able to leverage the NVLink between the GPU cards to reduce processing time to produce a significant increase in speed, especially when training. Finally comparing input-output data collected across each bus in real time revealed that speeds across the NVLink were extremely consistent whereas those across PCIe bus varied considerably from run to run due to competing processes running on the PCIe bus.

	AVIANN	Plankton	CASSA
DGX2	04:17	03:23	01:14
Power9	03:14	03:05	00:45
Supermicro	07:43	09:12	01:24

Table 2: Table showing run times for each of the tools discussed. Server configurations are listed in Table 1 and for this test each server had 4x NIVIDA V100 32G GPU for each of these runs. Each run in this table represent small runs designed to quickly show how NVLink changes our processing times. AVIANn data set was around classification of image data and was able to process 112827 images in the times listed. Plankton classification data sets contained 100 sets of 1850 images from specific regions of the ocean and each set was about 60MB in total size. The CASSA data was raw High Throughput Sequencing data and was a single input file of 2G (standard file size) and was aligned to the Human Genome.

In order to allow other researchers and computing groups to evaluate the effects of these architectures on performance, we have created a web-driven demonstration site that races the algorithms listed above on different hardware configurations. The site runs in real time and users can request access for a window of time allowing them to see multiple metrics including GPU load, GPU memory usage, GPU I/O and the actual command line output in real time (no video recording). Because we are providing this capability free of charge, we have only a limited set of hardware and can provide access to only a single group at a time. Therefore, users must request access and must run the system when their time slot becomes available. We invite groups and users to submit requests at the address http://aidemo.cgrb.oregonstate.edu.

This work was supported in part by TechData, a distributor of a very diverse set of computing technologies. TechData provided selected hardware help to ensure we could match equipment representing both architectures of the demo site.