CSN-221 Project

Submitted by,
Shashank Aital (19114076),
Batch O4,
The Department of Computer Science and Engineering,
Indian Institute of Technology, Roorkee.

Table of Contents

Hardware for Machine Learning

Before introducing any hardware, let us understand the need for special processors for Artificial Intelligence Fields like ML, DL, etc. Processors are an integral part of a computer. They are the brains that coordinate all the hardware and software required to get a computer working. The computers that we use in our daily lives need to do tasks over a large spectrum. From writing a report to listening to music, computers do everything for us. The processors used in these computers are thus, designed in such a way that they can perform a large range of tasks.
But, sometimes, we need application specific processors which excel at their specific jobs.

The Central Processing Unit (CPU)

The first CPU was developed by Intel in the early 1970s. (Pictured Below)

Most processors were designed with one core, i.e. it could perform only one operation at once. IBM released the first dual-core processor in 2001. Since then, more and more CPUs have been crammed into microprocessors: Some modern supercomputers can have more than 40!

Most of the computers that we use in our day to day life have a small number of cores. And this is enough for listening to music or surfing the internet. But for processing intensive tasks that ML poses, things can be a bit different.

- CPUs execute the instructions given to them sequentially.
- Linear algebra is the heart of Machine Learning and Deep Learning

- If we want to multiply, say 2 n x n matrices, our normal CPU would take n^3 cycles to process the output.
- Due to many IoT devices and boom in the technology and number of information providers, the size of data for ML / DL algorithms has also gotten a blast.
- The matrices used in DL Algorithms today can have thousands and thousands of parameters and performing operations on these matrices in $O(n^3)$ time could take years and years to train and infer a model.

Artificial intelligence: A challenge for processors?

Let us consider Deep Learning, a subfield of Artificial Intelligence. Deep Learning, as complicated as it may seem, is essentially a computer algorithm - a software construct. What we do in Deep Learning is, define an Artificial Neural Network in a programming language which would then be converted into a set of commands to run on a computer.

There are a few steps in making a Deep Learning Model. Few of them are:

- Preprocess input data
- Training the model
- Storing the trained model
- Deploy the model in production phase

Among all these steps, training a deep learning model is the most computationally expensive task.

Training a deep learning model

When we train a model, we perform the following two steps repetitively:

- The Forward Pass
- The Backward Pass

In the forward pass, the input is passed through the neural network and an output is generated. In the backward pass, we compare the output generated by the network to the ground truth and optimize the weights of the neural network accordingly. This is how a model is “trained”.

Now, we store the input as well as weights in matrices. Therefore, each of these above steps are matrix multiplications. This seems like a simple task. But, there are many of these simple tasks. For example, VGG16 (a convolutional neural network having 16 layers) has approximately *140 million parameters*, constituting the weights and biases. Now, multiplying a matrix this big would take ages on a traditional computer.

Inference

Inference is the process of using trained models to make predictions. This part involves only the forward propagation part discussed above and does not involve the processor intensive backward propagation step. Once a model is trained, it can be used in light weight devices with processors which are optimized for performance and power. **Habana Goya** is one of the processors used for the purpose of inference.

Solution

Data manipulation in ML / DL algorithms essentially involves two main operations:

1. Multiply a number of pairs of numbers
2. Add them together to get the result

As you may notice, the same above steps are used in matrix multiplication. These two operations are collectively called **Multiply-Accumulate Operations (MAC)**. These are the operations that are performed in a ML / DL Algorithm.

Now, we can see that most of these operations performed are independent of each other. That is, the input of any stage does not depend on the input of another stage. So, it is possible to get the same final output by executing all the operations simultaneously. This would literally reduce our processing time by a factor of the number of parallel processing units. Hence, **parallel processing** can be used in AI Algorithms.

The processors are judged based on their **throughput** and **latency**. More parallel processing directly increases throughput thus, training models faster. The internal architecture of the processor influences the latency of a processor. We try to reduce latency as far as possible. Lesser the latency, faster is the processor.

AI Accelerators

An **AI Accelerator** is a kind of specialised hardware accelerator or computer system created to accelerate artificial intelligence apps, particularly artificial neural networks, machine learning, robotics, and other data-intensive or sensor driven tasks. They usually have novel designs and typically focus on low-precision arithmetic, novel dataflow architectures or in-memory computing capability.

As the field of Deep Learning and Artificial Intelligence grew, specialized hardware units were designed or adapted from existing products to accelerate these tasks, and to have parallel high-throughput systems for workstations targeted at various applications, including neural network simulations.

Some of the most popular hardware accelerators are,

The Graphics Processing Unit (GPU)

GPU is a specialized chip that can do rapid processing, which was traditionally used for image rendering. They have become a key part of supercomputing. Modern GPUs have extremely parallel structure, making them more valuable in the computations involved in AI algorithms. Multiple GPUs are used on supercomputers, on workstations to train large DL models. In contrast to a traditional CPU, NVIDIA GPUs, contain chips that are known as CUDA cores and each one of these cores is a tiny processor that can execute some code. CPUs are designed to reduce latency (so that it can perform complex tasks in a small amount of time) whereas GPUs are designed to increase the throughput (so that it can perform multiple operations in a small amount of time).

The Vision Processing Unit (VPU)

A VPU is a rising class of microprocessor, and a particular type of AI accelerator intended to quicken machine vision tasks. These are designed to capture visual data from cameras and are built for parallel processing it. Some of these tools are low power and give high performance and may be plugged into user interfaces or any monitoring / security devices.

VPUs are fit for performing machine vision algorithms such as training CNNs (Convolutional Neural Networks), SIFT (Scale-Invariant Feature Transform) and other similar ones. They may include inbuilt camera buffers (bypassing any external buffers, thus, increasing operation speed) and have a greater emphasis on on-chip data flow between many parallel execution units.

The factors in favor of VPUs include the extensive use of smartphones and increasing adoption of edge AI, and expanding demand for advanced computing capacities for computer vision. One of the examples of VPUs is **Intel's Movidius Myraid X** which is being used in many edge devices. Target markets are robotics, the IoT, new classes of AR / VR / MR devices, integrated Machine Vision Acceleration into smartphones and other mobile devices.

Field-Programmable Gate Array (FPGA)

A Field-Programmable Gate Array is an integrated circuit (IC) made to be configured by a customer / a designer after manufacturing. FPGAs include a range of programmable logic blocks and a hierarchy of "reconfigurable interconnects" that enable the blocks to be connected together like many logic gates that can be inter-wired in various configurations.

Due to this feature of "reconfiguring", FPGAs can be more effective than GPUs in certain scenarios like, FPGAs can prove to be a lot faster than GPUs in terms of interface devices since they can be restructured to meet the needs of the interface device. GPUs are optimised for parallel processing of floating-point operations utilising thousands of small cores. But, FPGAs can do a wide range of logical functions simultaneously. This capability of FPGAs is being considered suitable for emerging technologies like self-driving cars or deep learning applications.

Today's FPGAs have big resources of logic gates and RAM blocks to implement complex data computations. FPGAs can be programmed to the appropriate structure to optimize the execution of various time intensive algorithms. This feature separates FPGAs from Application-Specific Integrated Circuits (ASICs), which are produced customly for certain tasks. Many hardware companies like **Xilinx** have launched their FPGA products as latest datacenter accelerators.

Application-Specific Integrated Circuit (ASIC)

These are highly sophisticated processors particularly designed while keeping the applications in which they will be used in mind. ASICs employ strategies such as optimised memory use, use of low precision arithmetic, etc. to increase the throughput of computation. Hardware acceleration is used to speed up the computation present in AI workflow.

For example, **Intel Nervana**, an ASIC offers a large amount of parallelism in server settings.

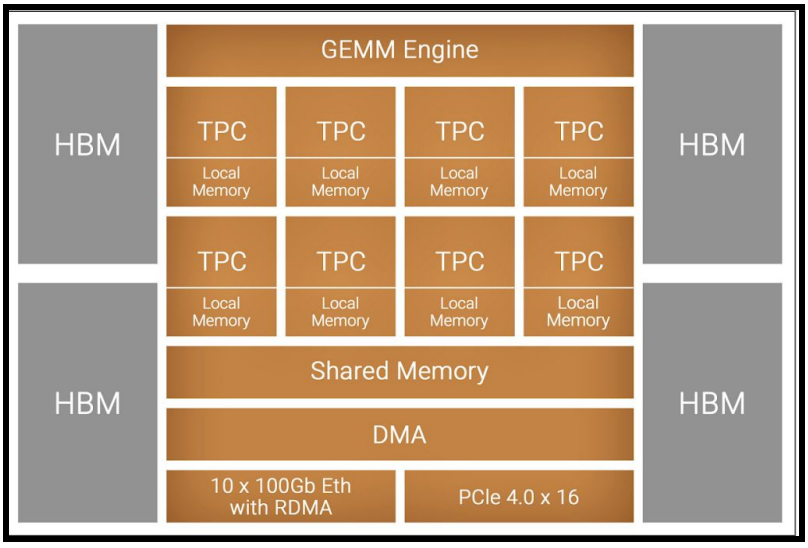
Intel - Habana

Intel bought out Habana Labs Ltd. for $2 billion. Habana has developed two AI chips namely, the **Habana Gaudi** and the **Habana Goya**. The former is a highly specialized neural network training chip, while the latter is a processor used for inference that uses neural networks in active deployments. Habana chips are far more powerful than Nervana chips (Nervana was also a part of Intel until it was dropped by intel when Habana came into picture). In recent benchmark tests, two Nervana chips were able to process upto 10,567 inputs per second in ResNet-50. But just one Habana Goya reached upto 14,451 inputs per second in the same test.

Habana Gaudi

Gaudi comprises a fully programmable Tensor Processing Core (TPC) cluster along with its associated development tools, libraries and compiler. The Gaudi HL-2000 chip is available in two card configuration types: a PCIe card and a Mezzanine Card. It integrates its processor with four High Bandwidth Memories (HBMs) in a single chip package giving it a memory bandwidth of **1 TB/s**.

High Level Architecture



- Gaudi uses a cluster of 8 TPC 2.0 cores.

- TPC 2.0 core is a VLIW SIMD vector processor with instruction set and hardware that were tailored to serve training workloads efficiently.
    - VLIW: Very Long Instruction Word Architecture
    - SIMD: Single Input Multiple Data
- TPC 2.0 core is C programmable. It has many workload-oriented features such as:
    - GEMM (GEneral Matrix to Matrix Multiplication) operation acceleration
    - Tensor addressing
    - Latency Hiding capabilities
    - Random Number Generation
    - Advanced Implementation of Special Functions
- Memory architecture:
    - on-die SRAM and local memories on each TPC
    - Four HBMs providing a total of 32 GB capacity and 1 TB/s bandwidth.
- Integrates RDMA (Remote Direct Memory Access) over Converged Ethernet engines. They generate a bidirectional throughput of 2 Tb/sec.

Training Algorithms

Two main strategies are used when training Deep Learning Models - Data Parallelism and Model Parallelism.

Data Parallelism

In Data Parallelism, every machine has a complete copy of the Deep Learning Model. Each machine receives a different portion of data, performs training locally, then transmits its parameter updates to the Parameter Server to share its training output with other machines.

When performing data parallelism, gradients can be sum-reduced in a hierarchical manner. For example, the first worker computes gradient tensor A, the second worker computes the gradient tensor B and so on. After the first reduction phase, half of the workers hold sum-reduced results of A+B and C+D. After the third step, a quarter of workers hold the sum-reduced result A+B+C+D. This reduction property of Gaudi Systems gives it large scalability.

Model Parallelism Training

In Model Parallelism, different machines are responsible for training different parts of the model. Implementing such parallelism requires low latency and high throughput.

Intel Xeon Family

Xeon is a family of processors manufactured by intel for non-consumer workstation, server and embedded system markets. For example, Google cloud has an option to use the processors from this family. Here are some of the features of the Skylake microarchitecture:

Skylake Mesh Architecture

In the previous versions of Xeon processors, ring based architectures were used for communication between the cores and memory units. The latest version introduces a mesh architecture which makes it easier to move information from one node to another and removing bottlenecks. The mesh architecture encompasses an array of vertical and horizontal paths thus, allowing traversal along the shortest path.

Intel Advanced Vector Extensions 512 (Intel AVX-512)

The new Xeon introduces new Intel AVX-512 instruction groups (AVX512BW and AVX512DQ) and a new capability (AVX512VL). These instructions enhance the performance of Xeon in vectorized operations, thus, increasing its performance in the applications of AI.

The AVX-512 ISA consists of several separate sets each having their unique CPUID feature bit. AVX-512 is an ISA made specifically for vectorizing operations. Some of them are:

F, CD, ER, PF

- **Foundation (F):** Expands 32-64 based AVX instructions to support 512-bit registers.
- **Conflict-Detection Instructions (CD):** Allows loops to be vectorized
- **Exponential and Reciprocal Instructions (ER):** Implement Transcendental operations
- **Prefetch Instructions (PF):** Prefetch the instructions

VL, DQ, BW

- **Vector Length Instructions (VL):** Extends AVX-512 to operate on XMM(128-bit) and YMM(256-bit) register formats

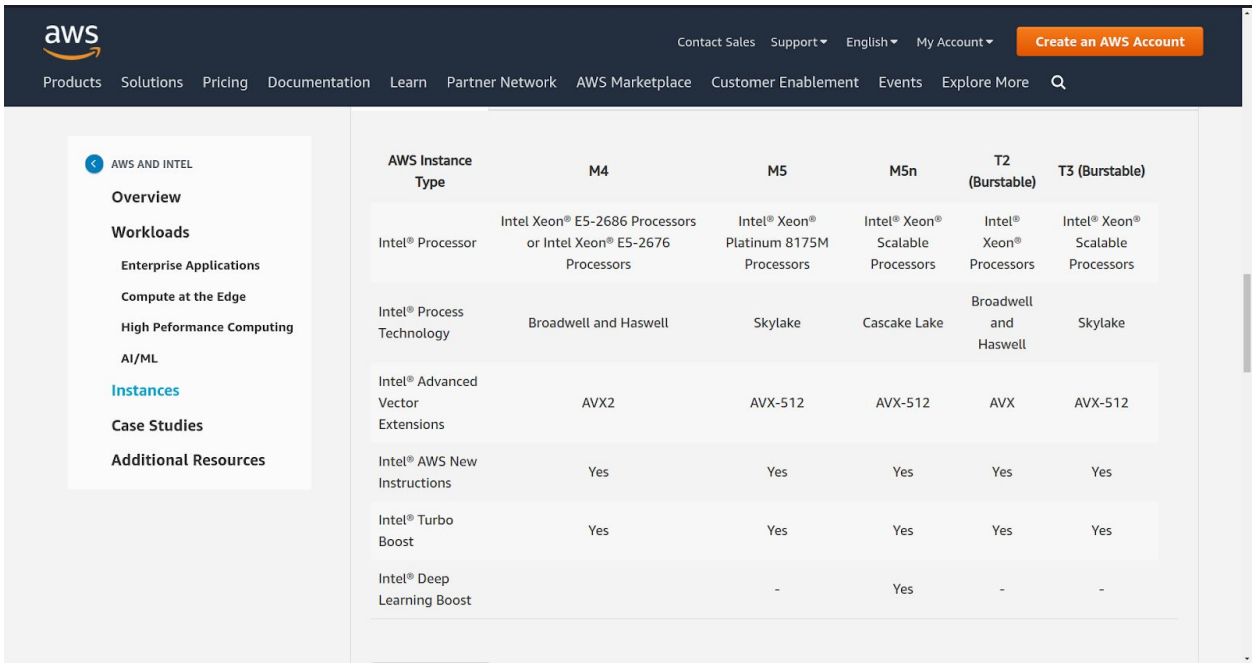
- **Doubleword and Quadword Instructions (DQ):** Adds new 32-bit and 64-bit AVX-512 instructions
- **Byte and Word Instructions (BW):** Covers 8-bit and 16-bit integer operations

4VNNIW, 4FMAPS

- **Vector Neural Network Instructions Word variable precision:** Vector instructions for deep learning, enhanced word, variable precision
- **Fused Multiply Accumulation Packed Single precision:** Vector instructions for deep learning, floating point, single precision

This ISA has many instructions that vectorize most of the operations that could be performed on an integer. Xeon incorporates these instructions on a processor level. Hence, the data is optimized for AI algorithms at a processing level.

Xeons are used on a large scale as scientific servers or cloud servers too. A major part of the DL community relies on the cloud for High Performance devices. Cloud services like Google Cloud, AWS, etc. use Xeons to do their bidding. Here are some of the xeon devices provided by AWS.



AWS Instance Type	M4	M5	M5n	T2 (Burstable)	T3 (Burstable)
Intel® Processor	Intel Xeon® E5-2686 Processors or Intel Xeon® E5-2676 Processors	Intel® Xeon® Platinum 8175M Processors	Intel® Xeon® Scalable Processors	Intel® Xeon® Processors	Intel® Xeon® Scalable Processors
Intel® Process Technology	Broadwell and Haswell	Skylake	Cascake Lake	Broadwell and Haswell	Skylake
Intel® Advanced Vector Extensions	AVX2	AVX-512	AVX-512	AVX	AVX-512
Intel® AWS New Instructions	Yes	Yes	Yes	Yes	Yes
Intel® Turbo Boost	Yes	Yes	Yes	Yes	Yes
Intel® Deep Learning Boost		-	Yes	-	-

Endianness: Little Endian

NVIDIA DGX-2 (AI Server)

DGX-2 is an AI server made by NVIDIA. It is a 2 petaFLOPS system made by integrating 16 **NVIDIA V100 Tensor Core GPUs.** It is powered by **NVIDIA DGX** software and the scalable architecture of **NVIDIA NVSwitch**.

System Specifications:

GPUs	16 * NVIDIA TESLA V100
GPU Memory	512 GB
Performance	2 petaFLOPS
NVIDIA CUDA Cores	81920
NVIDIA Tensor Cores	10240
CPU	Dual Intel Xeon Platinum 8168 @ 2.7GHz, 24-cores
OS	Ubuntu (Linux)

NVIDIA TESLA V100 (GPU)

Transistor Count: 21.1 billion

GPU Architecture: NVIDIA Volta

Volta is a microarchitecture developed by NVIDIA. Some of the details of Volta are as follows:

- Integration with APIs
    - Volta uses CUDA 9 for organizing groups of communicating threads.
    - Pascal and Volta include support for new cooperative launch APIs that support synchronization amongst CUDA thread blocks.
- Second generation NVIDIA NVLink

- Delivers high speed interconnect, higher bandwidth, more links, and improved scalability for multi-GPU system configurations.
- Volta GV100 supports up to six NVLink links and a total bandwidth of 300GB/sec.

Memory

- HBM2 Memory: Faster, Higher Efficiency
  - 16 GB HBM2 memory subsystem delivers 900 GB/sec peak memory bandwidth.

Other specifications

NVIDIA Tensor Cores	640
NVIDIA CUDA Cores	5120
Double-Precision Performance	8.2 TFLOPS
Single-Precision Performance	16.4 TFLOPS
Tensor Performance	130 TFLOPS
GPU Memory	32 GB
Memory Bandwidth	1134 GB/sec

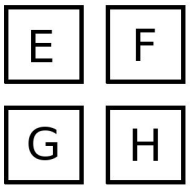
Due to the tensor cores, amazing parallelism is achieved and hence, DGX-2 is one of the most powerful systems used for AI computations.

Google TPU

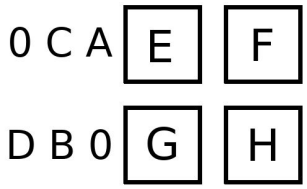
- Most of the operations in ML are performed by multiplying and then adding. Let's call this a single **Multiply-Accumulate(MAC)** operation.
- Now, CPU is a sequential machine i.e. it takes one instruction, processes it and then takes the next instruction. Hence, it is a scalar machine.
- GPUs, on the other hand, take a 1D array of operands and perform operations on them simultaneously. They are vector machines.
- But, in deep learning, we have matrix multiplications. Literally a lot of matrix multiplications. So, to optimize this, we build a matrix machine. We will focus only on MACs that perform Matrix multiplication.
- TPUs achieve high performance by using **systolic arrays.**
    - A systolic array is a hardware algorithm. It describes the pattern of cells on a chip that compute matrix multiplication. Consider the following matrix multiplication:

$$\begin{pmatrix} A & B \\ C & D \end{pmatrix} \times \begin{pmatrix} E & F \\ G & H \end{pmatrix} = \begin{pmatrix} AE+BG & AF+BH \\ CE+DG & CF+DH \end{pmatrix}$$

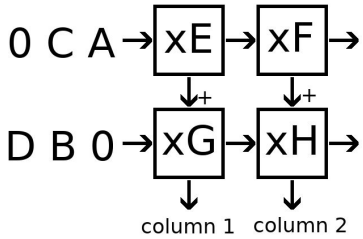
    - We implement matrix multiplication by storing the operands in a grid. This is the actual hardware grid that is used in TPUs and is as big as 128 x 128.
    - Say A B | C D are the activations and E F | G H are the weights. First, we load our weights into the grid:



    - Then, our activations go into an input queue. For example,



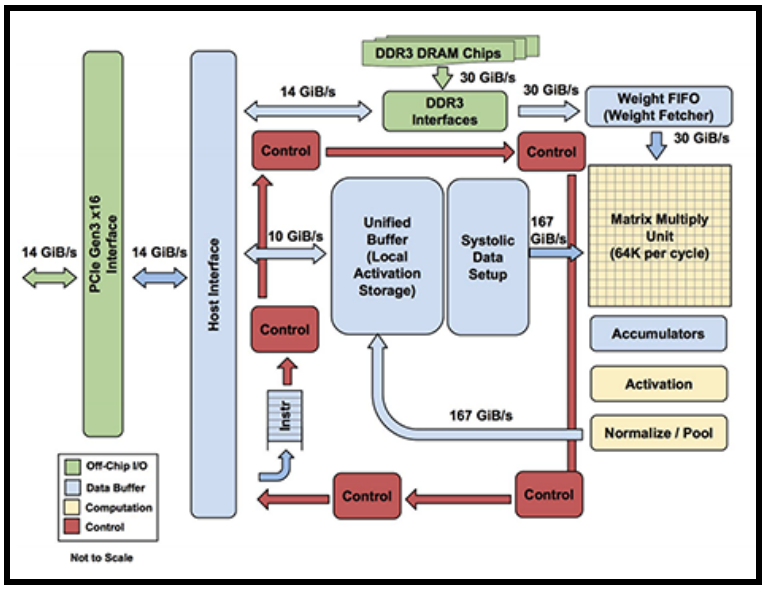
- Then, on every clock edge, this is what happens in each cell simultaneously:
  - Multiply weight in the cell and activation coming from the left. If there is no cell on the left, take the input from the input queue.
  - Add the product to the partial sum coming from the cell above. If there is no cell above, take the partial sum equal to zero.
  - Pass the activation to the cell to the right. If there is no cell to the right, throw away the activation.
  - Pass the partial sum to the cell at the bottom. If there is no cell at the bottom, store the partial sum as output.
- Let's look into what's happening in our 2 x 2 example:



  - Cycle 1:
    - Input A is loaded to the cell X_{11} and the product AE is calculated.
    - AE is added to the partial sum from above (i.e. 0).
    - Activation A is passed to the cell X_{12}.
    - Partial product is passed to the cell X_{21}.
    - For all the other cells, all the activations are zero in this cycle.
  - Cycle 2:
    - Cell X_{11}:
      - The input activation C is loaded into X_{11}.
      - It is multiplied by E to get CE. It is added to the partial sum (0) to get CE.
      - C is passed to X_{12} and CE is passed to X_{21}.
    - Cell X_{12}:
      - The cell X_{12} receives the input activation from X_{11} i.e. A.

        - A is multiplied by F to get AF. It is added to the partial product from above (i.e. 0) to obtain AF.
        - This AF is passed to the cell below (X_{22}).
        - Also, activation A is passed to the right. Since there is no cell, the input is discarded.
    - Cell X_{21}:
        - The input B is received from the input queue.
        - It is multiplied by G to get BG. The product BG is added to the partial sum AE (received from X_{11} in the previous clock cycle) to obtain (AE + BG).
        - The activation B is passed to X_{22} and the partial sum is passed as output (as there is no cell below). This gives us the first element of the first column of our answer
    - This process is continued for a total of 8 cycles to get the complete output of multiplying our activations and weights.
- Please note that the input queues are appended with zeros in order to ensure that they enter the correct cell at the correct moment.
- This systolic array multiplies 2 matrices in a staggering (3n-2) cycles. Whereas the traditional sequential processor requires n^3 cycles.
- Hence, the core advantage of the TPU is its MXU - a systolic array matrix multiplication unit.

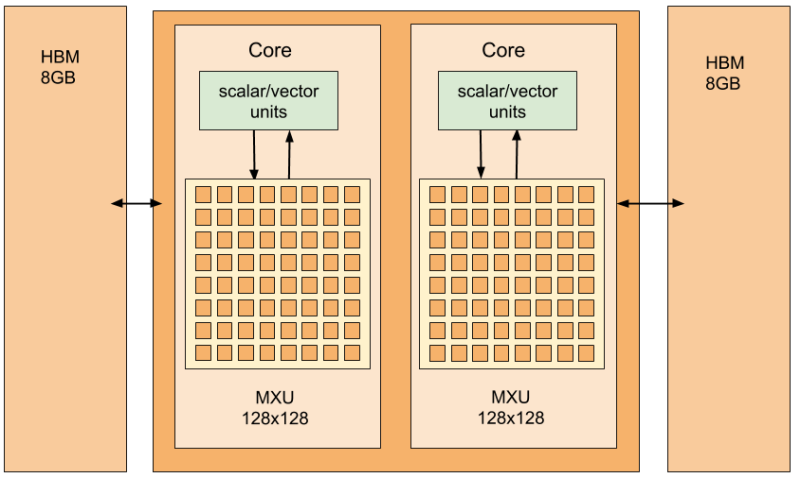
- Now, let's look into the other parts of the TPU and its working:



- When the chip starts, the buffers and the DDR3 RAM are emptied.

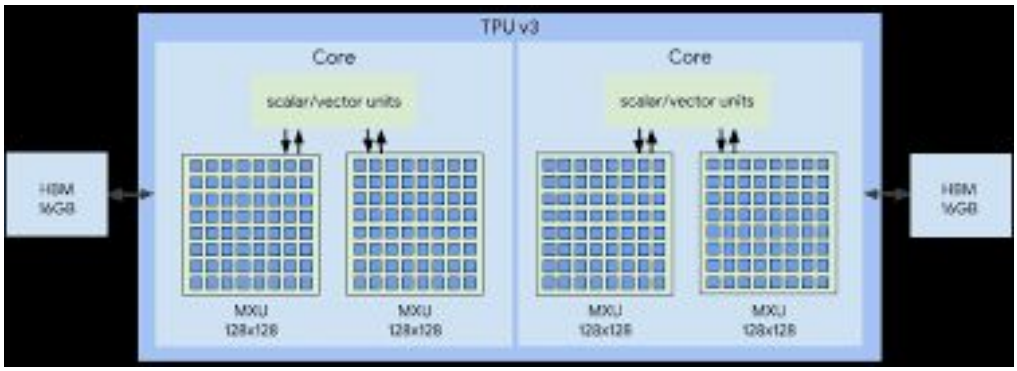
- When the user loads the TPU compiled model, the initialized weights are stored in the DDR3 memory.
- Then, the host fills the activation buffer with input values.
- Then, a control signal is sent to the weight fetcher (FIFO) to load the MXU with weights for this layer.
- The host triggers the execution of MXU. The activations propagate through the MXU and the final results are stored in the accumulators.
- Then, they run through the activation pipeline (which applies standard activations to the outputs like ReLU, dropout, etc.)
- Simultaneously, the next layer is placed in the buffer. The activation output of the current layer serves as the input for the next layer.
- The above steps are repeated for all the layers.
- Activations of the last layer are sent back to the host.

- In TPUv1, there was only one MXU (only one core). But the new TPUs have upgraded features:
  - TPU v2:



    - 8 GiB of HBM for each TPU Core
      - HBM gives a higher bandwidth than DDR3 and hence, is a lot faster.
    - One MXU for each TPU core
    - Upto 512 total TPU cores and 4 TiB of total memory in a TPU Pod

- TPU v3:



  - 16 GiB of HBM for each Core
  - Two MXUs for each TPU core
  - Upto 2048 total TPU cores and 32 TiB of total memory in a TPU Pod.

- We can access Google TPUs via **Google Cloud** or **Google Colab**.

Comparison Chart for AI Accelerators

Chip	Habana Gaudi	NVIDIA Volta	TPU v1	TPU v2	TPU v3
Power (W)	300	300	75	200	200
Die Size (nm)	500	815	28	-	-
Peak (TFLOPS)	15,453 images/sec (RESNET-50)	125	23 (INT16)	45	90
Memory Bandwidth	1000 (HBM2)	900(HBM2)	30(DDR3)	600(HBM)	1200(HBM2)
IO (GB/s)	250	300	14	8	8

Conclusion

In this project, I learned many things about how algorithms are implemented in real life devices. I have completed a specialization in Deep Learning. But I never really realized that the field of architecture required to develop DL Algorithms had developed so much. For me, major DL Algorithms were only to be seen and not executed. Due to this assignment, I learned about many frameworks and cloud services that one can use for training / testing his/her model.

This project gave me a chance to explore various processors and their architectures which are made specially to incorporate DL Algorithms. I could understand the design of those processors and ISAs and the roles that they played in reducing the training time. I got a chance to read about many innovations in the field like Intel Habana. Due to the research work I put in the project, I came to know about different computer devices like HBM. I came to know about how the matrices are represented in a TPU and how instructions are performed on them. Before this project, I used Google TPUs as a blackbox but now, I understand the inner workings of the TPU. In this project, the research was not limited only to the 4 processors I discussed. After reading white papers for devices, there were many terms that I was unfamiliar with. Googling about those terms led to introduction of more unfamiliar terms. So, as an end result, I was equipped with knowledge of many concepts.

This project has not only helped me in learning new things but also helped me revise my concepts. It has amplified my interest in not only the field of Computer Architecture, but also, reignited my interest in the field of Deep Learning. I am certainly looking forward to such informative assignments and projects.

References

- https://www.elprocus.com/what-are-different-types-of-processors-applications-and-characteristics/#:~:text=There%20are%20five%20types%20of,Processor%2C%20DSP%20and%20Media%20Processor.
- https://www.geeksforgeeks.org/difference-between-normal-processor-and-ai-processor/
- https://algorithmia.com/blog/hardware-for-machine-learning
- https://analyticsindiamag.com/the-different-types-of-hardware-ai-accelerators/
- http://www.cs.cornell.edu/courses/cs4787/2019sp/notes/lecture25.pdf
- https://medium.com/@shan.tang.g/a-list-of-chip-ip-for-deep-learning-48d05f1759ae
- https://www.intel.com/content/www/us/en/artificial-intelligence/hardware.html
- https://software.intel.com/content/www/us/en/develop/articles/intel-xeon-processor-scalable-family-technical-overview.html
- https://medium.com/@antonpaquin/whats-inside-a-tpu-c013eb51973e
- https://cloud.google.com/tpu/docs/system-architecture
- https://habana.ai/wp-content/uploads/2019/06/Habana-Gaudi-Training-Platform-whitepaper.pdf
- https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/dgx-1/dgx-2-datasheet-us-nvidia-955420-r2-web-new.pdf
- https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf
- https://www.nvidia.com/en-in/data-center/v100/