ML Acceleration on GPUs

GPU and ML

Learning Objectives

1. Explain the basics operations of machine learning workloads
2. Describe the benefits of GPU for ML workloads

Why are GPUs Good for ML?

1. High number of floating-point operations
2. High data parallel operations
3. GPUs have dense floating-point operations
4. High memory bandwidth (e.g., GDDR, HBM)
5. Flexible data format standards (e.g., TF, BF, INT4, etc.)
6. Statistical computing based computations

DNN Operation Categories

1. Elementwise operations
   - Activation opertaions
2. Reduction operations
   - Pooling operations
3. Dot-product operations
   - Convolution operations, GEMM

Background of GEMM

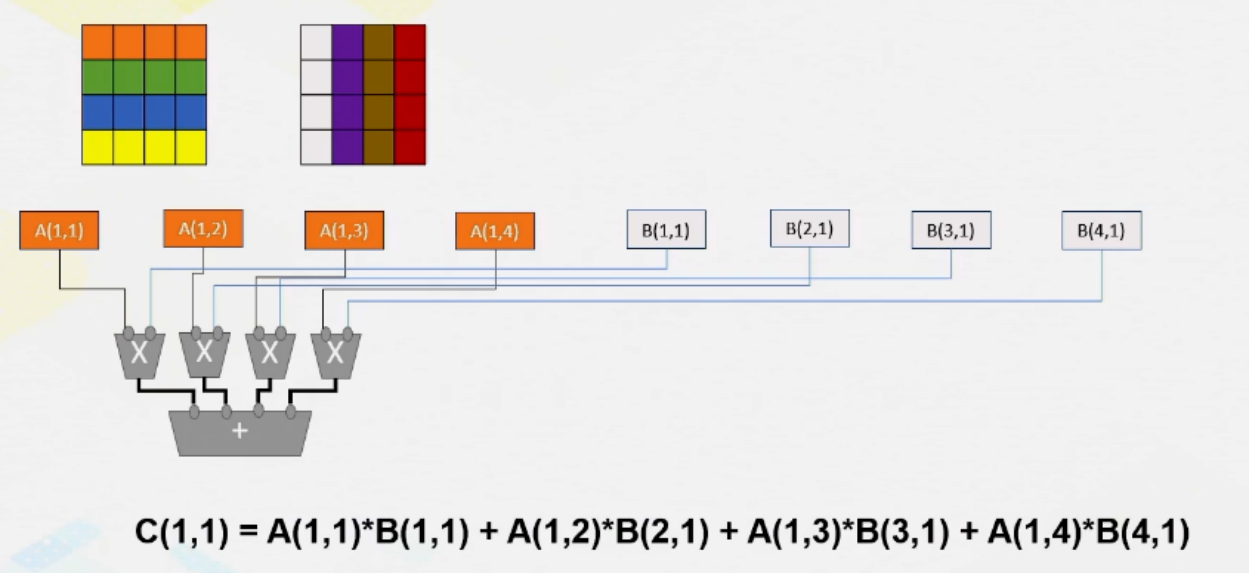
1. General Matrix Multiplications (GEMM)
   - C = a AB + bC
   - A, B, and C are mxk, kxn, and mxn matrix
   - a = 1 and b = 0 becomes C = AB
2. Popular in fully-connected layers, convolution layers
3. Production of A and B -> M * N * K fused multiply-adds (FMAs)
   - 2 * M * N * K flops
   - FP16 inputs FP32 accumulator
4. Arithmetic intensity = number of flops / number of byte accesses
   - 2(MNK) / 2(MK + NK + MN)
5. Use arithmetic intensity to compare the machine's FLOPS/B

Difference Between SIMD vs Tensor

1. 4x4 matrix computation
   - 4 threads and each thread repeats a loop
2. Sum needs to be stored in a register
   - Since NVIDIA thread is private, one thread needs to perform accumulation
   - If 16 threads are performing the work in parallel, the sum variable needs a reduction mechanism

SIMD/SIMT Operations of Matrix Multiplications

1. 16 SIMD/SIMT operations are needed for 4x4 matrix multiplications

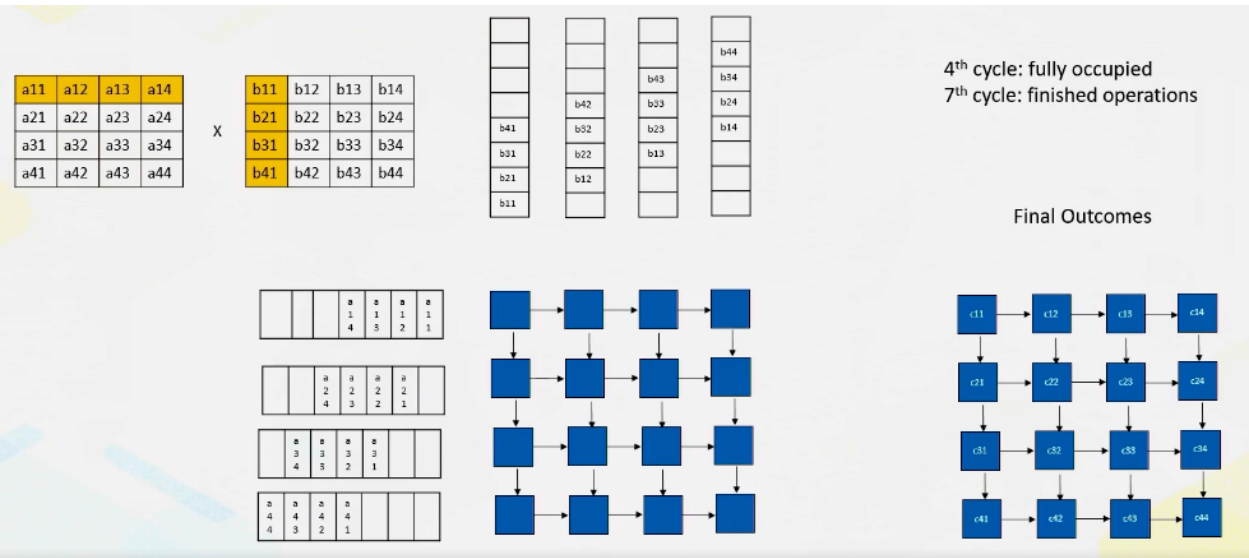


SIMD Operations

Tensor Cores

1. Tensor cores perform matrix multiply and accumulate (MMA) calculations
2. Hundreds of tensor cores operating in parallel in one NVIDIA GPU enable massive increses in throughput and efficiency
3. Support sparsity
4. Each A100 tensor core can execute 256 FP16 FMA
5. INT8, INT4, and binary 1-bit predictions added

Matrix Multiplications with Systolic Arrays



Systolic Arrays

Transformer Engine

1. Introduced in GH architecture
2. To accelerate transfer layers
3. Transformer engine dynamically scalres tensor data into representable range
4. FP8 operations
5. Bring a chunk of data efficiently to utilize tensor units
6. TMA address generation using copy descriptor

Summary

1. GPU accelerates matrix multiplications in ML workloads
2. SIMD/SIMT accelerates but dedicated matrix multiplication units improve the performance even more
3. Fully utilizing memory bandwidth is important

Floating Point Formats

Learning Objectives

1. Describe IEEE floating-point formats
2. Explain the benefits of quantization

FP16 vs FP32

1. FP16 uses half the precision of FP32
2. IEEE standards have 32 bits/64 bits (single/double precisions)
3. Recap: Producation of A and B -> MNK fused multiply-adds -> 2MNK FLOPS
    - FP16 inputs into FP32 accumulator
        - Arithmetic Intensity = 2MNK / 2(MK + NK + MN)
    - FP8 inputs into FP32 accumulator
        - Arithmetic Intensity = 2MNK / (MK + NK + MN)
4. Changing the input FP formats can change the arithmetic intensity significantly

Benefits of Quantization

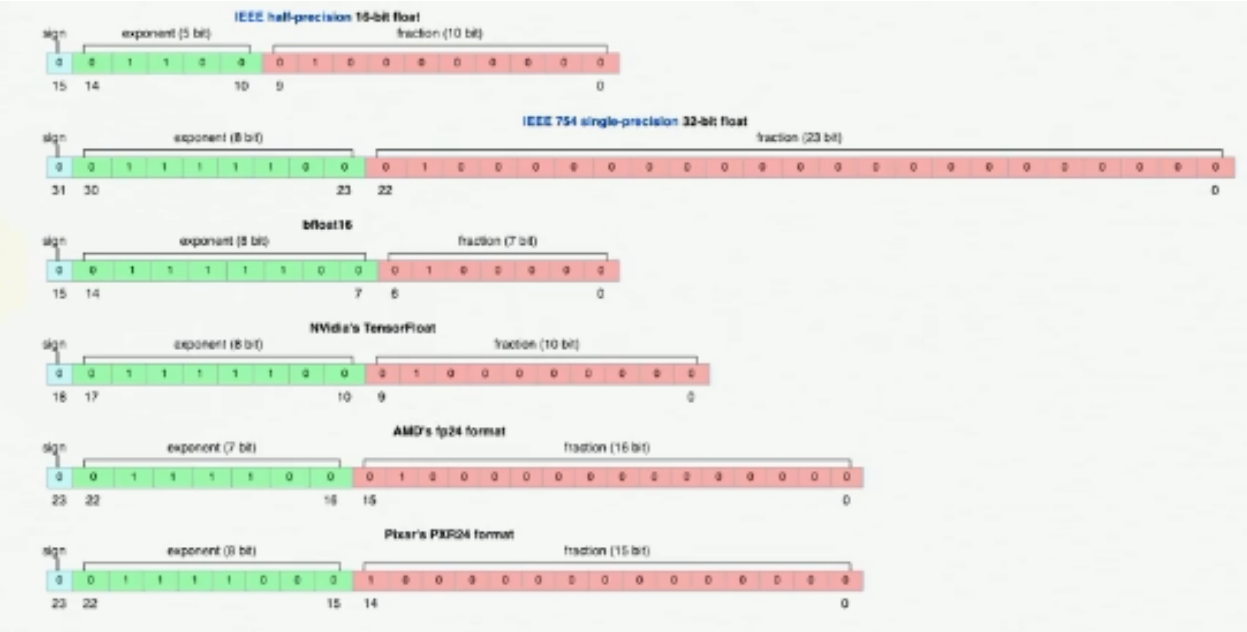
1. Reduce the storage size
2. Increase the arithmetic intensity
3. Advanced floating-point operations also increase the throughput
4. Ideally, the performance of FP8 is 2x of FP16 operations

Floating Point Basics

1. Representing 0.0012 = 1.2e-3
2. Floating point is represented with fraction (e.g. 1.0) times exponent (10^-3)
3. Since we are representing the numbers in binary, fraction and exponents are base 2
4. Split between sign bit, exponent bits, and fraction bits

Different Floating Point Formats

1. Different number of fraction bits and exponent bits
2. More exponent bits cover a wider range and more fraction bits cover more precision

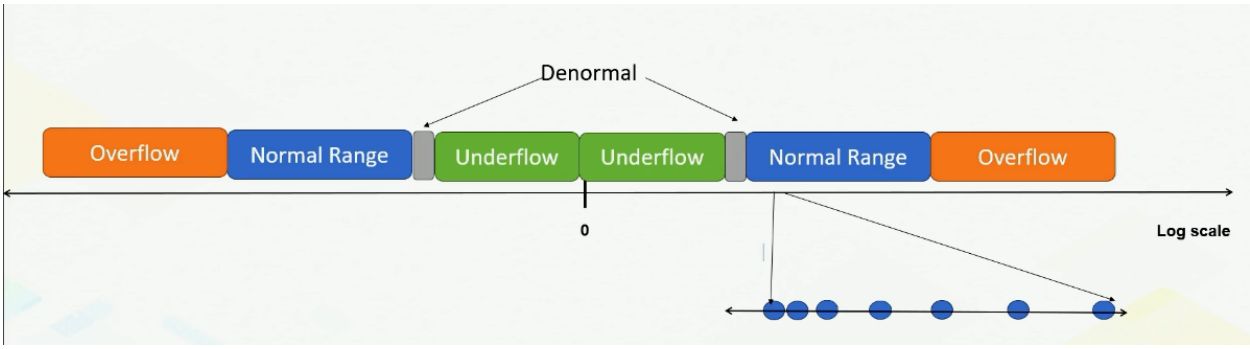


Floating Point Formats

Some More Details

1. Exponent values can be both positive and negative
2. Floating points need to represent 2^(-exp1) to 2^(exp2)
3. Negative exponent values represent less than 1; positive exponent values represent greater than 1
4. Sign bit is to indicate negative or positive value of actual floating point
5. How do we represent exponent values without sign bits?
    - Solution: Offset the value
    - Subnormal numbers (denormalized numbers): When all exponents are 0

Floating Point Number Representations



Floating Point Representation

IEEE 754 Formats

1. Formats (sign, exponent, and mantissa)
    - Special values (positive/negative zero, positive and negative infinite values, and NaN)
2. Denormalzed values (subnormal values)

3. Rounding modes

Quantization

1. Reduce the number of required operations
2. Commonly used for reducing input operations while keeping the accumulator in high precision
3. Quantization often leads to a non-uniform quantization
4. Sometimes, value transformations are used to overcome non-uniform quantization
    - e.g., shifted and squeezed 8-bit format (S2FP8)

Summary

1. Quantization can improve the arithmetic intensity and reduce data movement cost
2. Quantization can also improve the efficiency of computations

Tensor Core

Learning Objectives

1. Describe the architecture design methods for ML accelerator components on GPUs
2. Explain the main design criteria

Designing ML Accelerator Units on GPUs

1. Consideration factors
    - What functionality?
    - Benefits over the existing GPUs
2. Compute unit design and the scale
3. Data storage and movement

Common Steps of Designing Accelerators

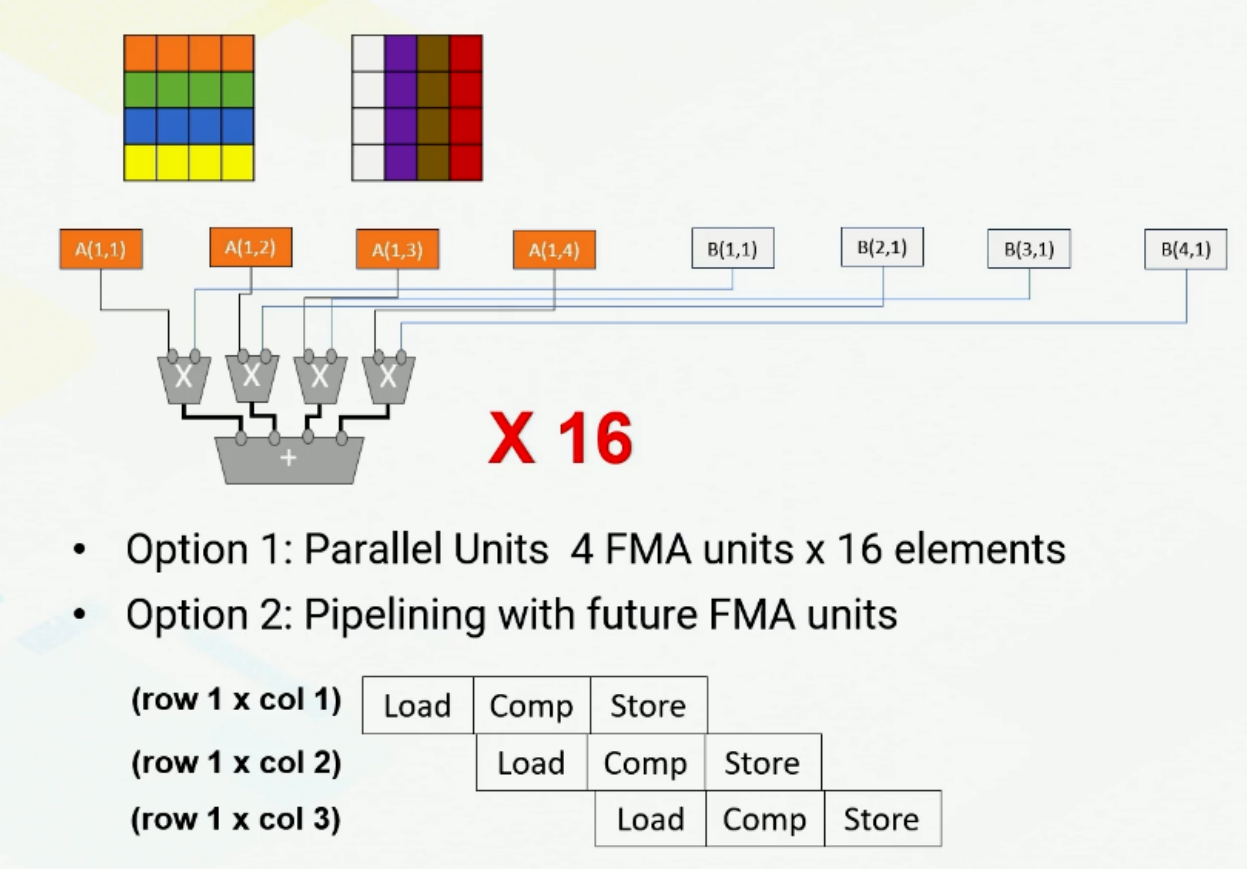
1. Identify frequently executed operations
2. Performance benefit estimation
    - Software approach vs hardware approach; software approach is using existing ISAs
3. Design interface and programmability
    - What ISA to add?
    - What storages?
    - Separate registers or shared registers?
4. Consider the possibility of combining multiple features
    - Any other operations to combine?

Matrix Multiplication Accumulator

1. Most commonly used in ML workloads
2. SIMD operations still require row-by-row operations
3. Large matrix multiplication units can be implemented with systolic arrays
4. Design decisions: many small matrix multiplication units vs a large matrix multiplication unit?
5. Area and programmability choices
6. NVIDIA started with 4x4x4 matrix multiplication units
7. Intel's AMX (Advanced Matrix Extensions) support 16x16 matrices

Possible Matrix Multiplication Unit Design

1. Option 1: Parallel Units 4 FMA units x 16 elements
2. Option 2: Pipelining wiht future FMA units



Matrix Multiplication Unit Design

Data Storage and Movement

1. Input matrices have to come from memory
2. Design decisions: dedicated registers or shared memory
3. 4x4x4 matrix operations require at least 3x16 register space
4. NVIDIA: registers are private to threads
5. Memory space can be used for storing tensor data
6. Eventually, data needs to come from memory
7. In NVIDIA, with tensor core, new asynchronous copy instruction is introduced: loads data directly from global memory into shread memory, optionally bypassing L1 cache, and eliminating the need for intermediate register file (RF) usage
8. New aynchronous barrier instructions also introduced

Supporting Sparse Operations

1. Sparse Matrix Multiplication: some elements are zero values
2. Widely used in high-performance computing programs
3. Software approaches: use compressed data format
4. Instead of storing all values, only store non-zero values
   - (row index, column index, value)
5. With pruning operations, sparsity increases

Supporting Sparsity in Hardware

1. Structured sparsity: assume only a fixed % of elements are non-zeros
    - Assumption of 50% sparsity (use 1 bit to indicate left or right position)
2. With a structured sparsity, storing index information is simplified
3. Accelerating SpMV (Sparse Matrix-Vector Multiplication) is a big research topic
4. Reduce storage space and improve throughput

Sparse Matrix Multiplication Hardware

Summary

1. Reviewed design trade-offs of designing ML accelerators on GPUs
2. Review GEMM and SPMV operations