EE 7722—GPU Microarchitecture

EE 7722—GPU Microarchitecture

URL: `https://www.ece.lsu.edu/gp/`.

Offered by:

David M. Koppelman
3316R P.F. Taylor Hall, 578-5482, koppel@ece.lsu.edu, https://www.ece.lsu.edu/koppel
Office Hours: Monday - Friday, 15:00-16:00.

Prerequisites By Topic:

- Computer architecture.
- C++ and machine language programming.

Text

Papers, technical reports, etc. (Available online.)

Course Objectives

- Understand low-level *accelerator* (for scientific and ML workloads) organizations.
- Be able to fine-tune accelerator codes based on this low-level knowledge.
- Understand issues and ideas being discussed for the next generation of accelerators . . .
  . . . including tensor processing (machine-learning, inference, training) accelerators.

Course Topics

- Performance Limiters (Floating Point, Bandwidth, etc.)
- Parallelism Fundamentals
- Distinction between processor types: CPU, many-core CPU, (many-thread) GPU, FPGA, large spatial processors, large systolic array.
- GPU Architecture and CUDA Programming
- GPU Microarchitecture (Low-Level Organization)
- Tuning based on machine instruction analysis.
- Tensor Processing Units, machine learning accelerators.

Graded Material

Midterm Exam, 35%

Fifty minutes in-class or take-home.

Final Exam, 35%

Two hours in-class or take-home.

Yes, it's cumulative.

Homework, 30%

Written and computer assignments.

Lowest grade or unsubmitted assignment dropped.

Course Usefulness

Material in Course Needed For:

- Those designing the next generation of accelerator chips.
- Those writing high-performance scientific programs.
- Those writing high-performance machine-learning engines.
- Those writing high-performance graphics programs.
- Those interested in future computer architectures.
- Compiler writers.

Course Resources

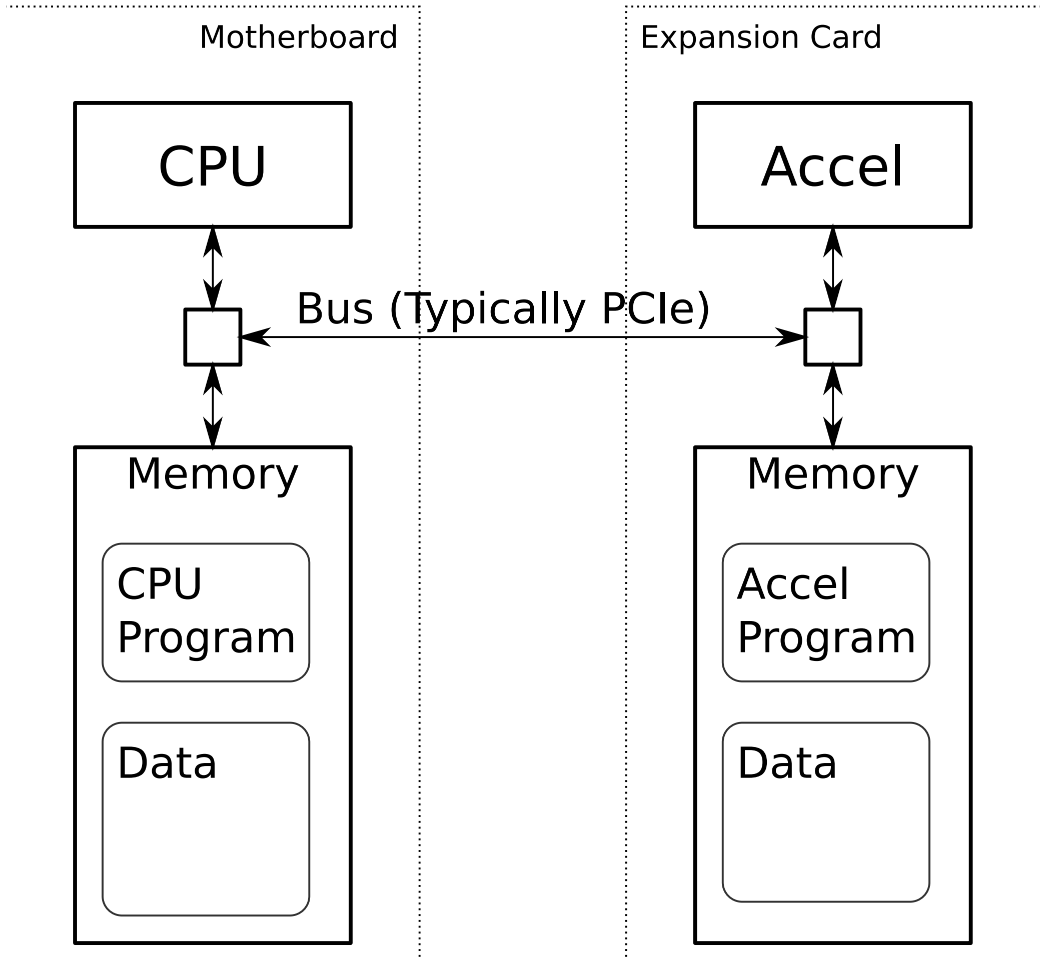
- Slides and other material via `https://www.ece.lsu.edu/gp/`
- Code examples in git repository `git://dmk.ece.lsu.edu/gp`
- Web site also has homework assignments, exams, grades, and other material.
- Announcements are on course home page and available as a Web (RSS) Feed.

Accelerator:

A specialized processor designed to work alongside a general-purpose processor (CPU).

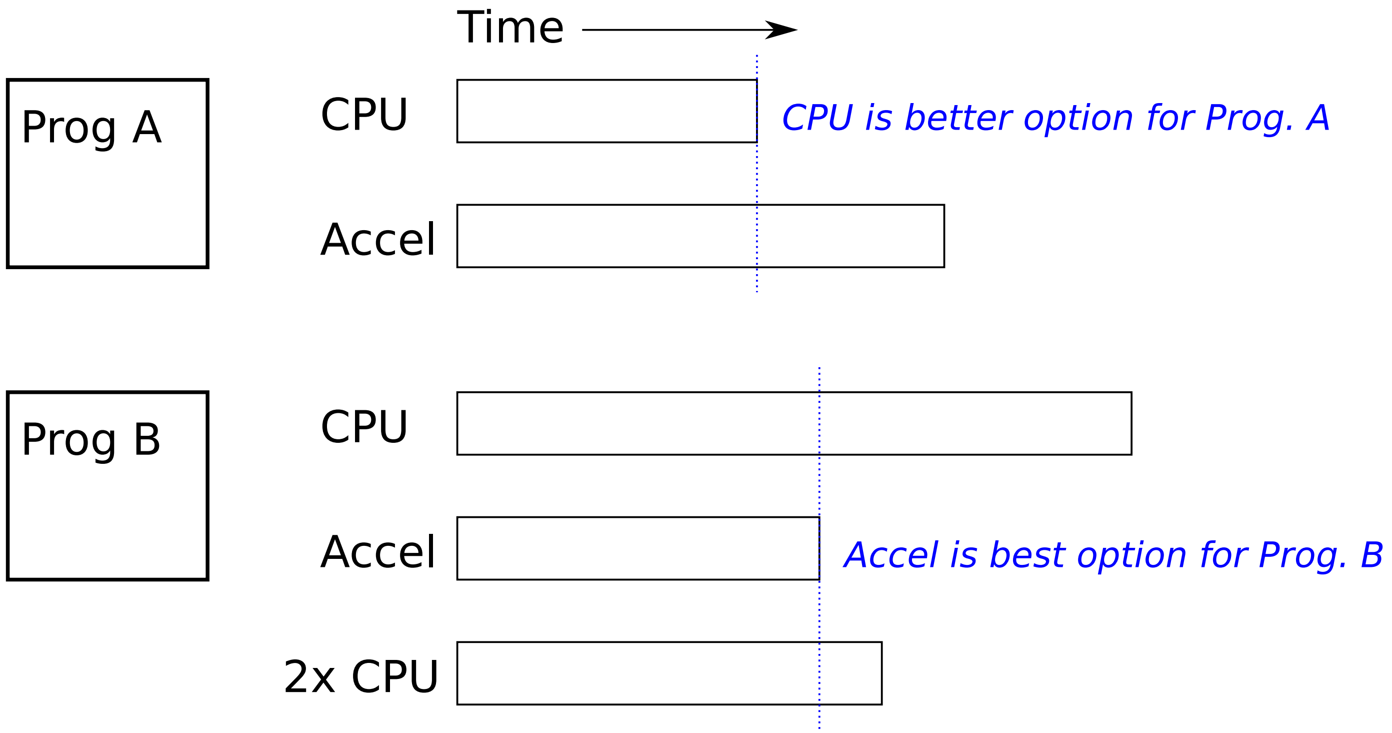
Work is split between the different devices . . .

. . . each does what it's best at.



Accelerator Benefit

Accelerator can execute some code faster than CPU.


Time
Prog A
CPU
Accel
CPU is better option for Prog. A
Prog B
CPU
Accel
Accel is best option for Prog. B
2x CPU

Program B is faster on the accelerator . . .
. . . but for Program A the accelerator hurts performance.

For Program B, two CPUs almost as good as one CPU plus one accelerator.

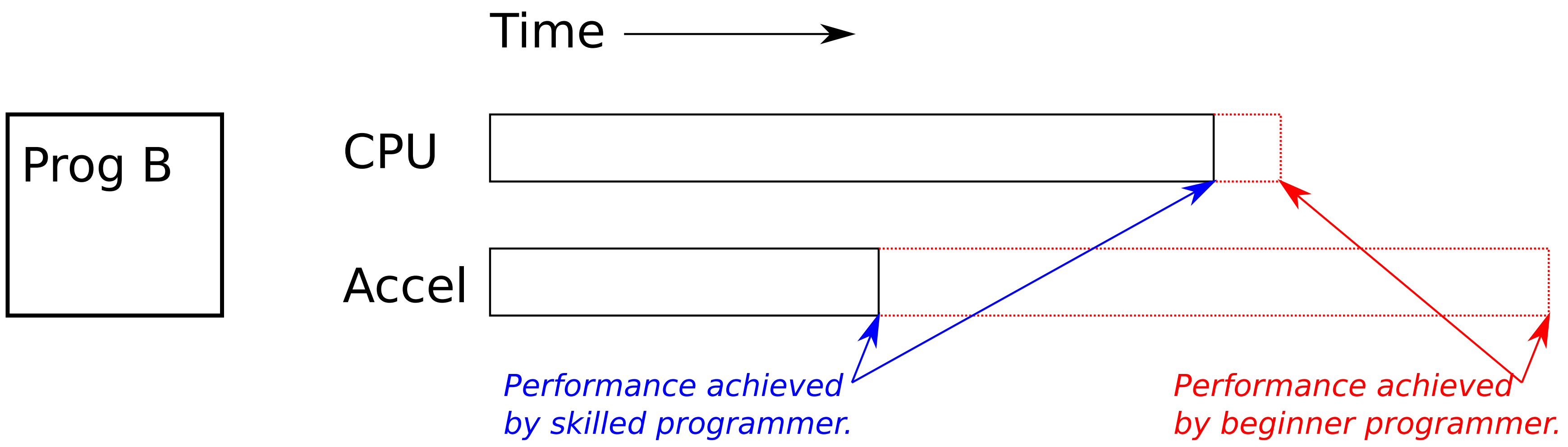
Accelerator Programming Challenge

CPUs are more forgiving.

Not paying attention to things may cost a few percent.

GPUs must be programmed with care.

Not paying attention to things can result in much worse performance.


Time
Prog B
CPU
Accel
Performance achieved by skilled programmer.
Performance achieved by beginner programmer.

Common Accelerator Types

- *GPU (Graphics Processing Unit)* — *E.g.*, NVIDIA Ampere A100 [3]
- *Many-Core Processor* — *E.g.*, Intel Xeon Phi (Discontinued)
- *Large Systolic Array Tensor Processor* — *E.g.*, Google TPU3 [2]
- *FPGA Accelerator* — *E.g.*, Nallatech PCIe-180
- *Processor-In-Memory Digital Accelerator* — Research area.
- *Processor-In-Memory ReRAM Accelerator* — Research area.

Example:

> “Our computer was taking four days to compute the 48-hour forecast, so we bought a system with 3 accelerators: an NVIDIA K20c, a Xeon Phi, and a Nallatech board, all of these were given work that would have been performed by the general-purpose CPU, an Intel i7.”

GPU (Graphics Processing Unit):

A processor designed to execute a class of programs that includes 3D graphics and scientific computation using a large number of threads.

A Brief History

GPUs originally designed *only* for 3D graphics [1].

Large economies of scale made them cheap.

Resourceful scientific users disguised their work as 3D graphics.

GPU makers started supporting scientific and other non-graphical work.

GPU evolved into a second kind of processor, with 3D graphics just one application.

Current Status

Always used in systems needing 3D graphics, from cell phones to workstations.

Often used for scientific computation …
… but acceptance has been slow due to difficulty of porting code.

GPU Product Examples

- NVIDIA RTX 4090 — High-end GPU for home use.
- NVIDIA Hopewell H100 — High-end GPU for non-graphical computing…
  … including half-precision FP support for deep learning applications.
- AMD Radeon R9 — High-end GPU for home use.

Many-Core Processor:

A processor designed to execute a class of programs that includes 3D graphics and scientific computation using simple cores and wide vector units.

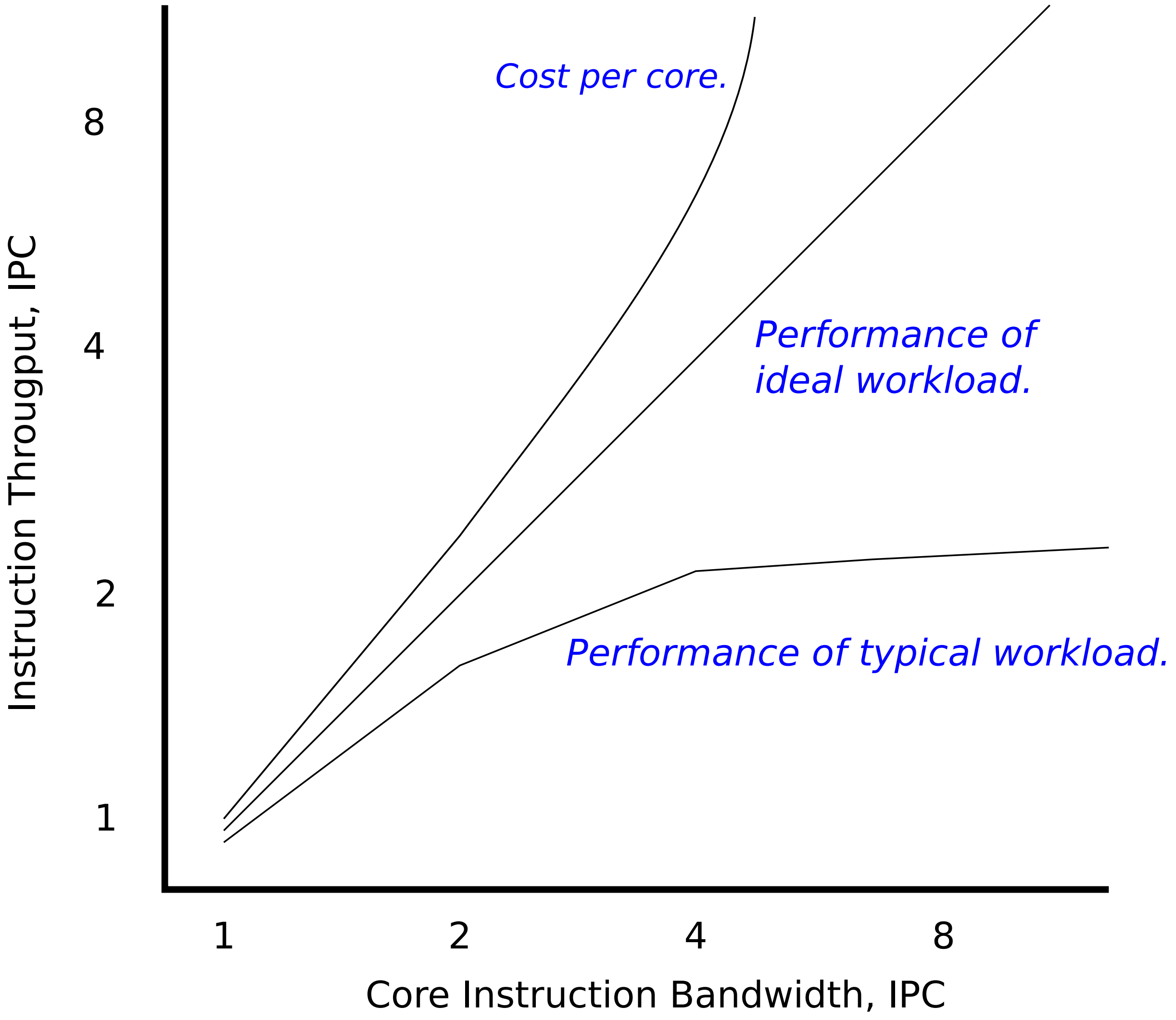
Motivation

Larger cores. . .

. . . are expensive (w.r.t. ideal perf.). . .

. . . and slow (w.r.t. ideal perf.)

So, use lots of small cores.



A Brief History of Many-Core Accelerators

Long known that peak performance of small-core chip > large-core chip of same area…
… the problem was parallelization.

Many research and one-off designs used chips filled with simple cores.

The inclusion of wide vector units meant few cores would be needed.

Idea used by Intel for a graphics chip, project Larrabbe.

Larrabbe re-targeted at scientific computing, product named Phi.

Current Status

Major commercial product, Phi, discontinued.

So, it's back to being just a research idea, but one that won't go away.

Many-Core Processor Examples

- Intel Xeon Phi — For scientific computing. Discontinued.
- Sun T2000 — Meant for server workloads.

Tensor Processing Unit

Tensor:

A linear mapping from one object to another. A common example is an $m \times n$ matrix, which maps n-element column vectors to m-element column vectors: $v = M \times u$.

For this class, a tensor is a multi-dimensional array.

Tensor Processing:

Transforming tensors and applying operations on the resulting tensor.

For the most part tensor processing is a generalization of matrix multiplication:

```
for ( int n=0; n<N; n++ )
  for ( int m=0; m<M; m++ )
    for ( int x=0; x<X; x++ )
      for ( int y=0; y<Y; y++ )
        for ( int c=0; c<C; c++ )
          for ( int i=0; i<X; i++ )
            for ( int j=0; j<Y; j++ )
              oput[n][m][x][y] += iput[n][c][i][j] * w[m][c][x][y][i][j];
```

Tensor Processing Unit:
Hardware that can efficiently perform tensor processing

The term is used to describe:

Specialized, large systolic-array accelerators, including Google's TPUs.

Sometimes it can refer to a functional unit . . .
. . . in a processor designed for more than tensor processing.

TPU Motivation

Can use lower-precision arithmetic.

Data use and re-use patterns.

High concentration of multiply/add (accumulate) operations . . .
. . . forming matrix/matrix multiply or matrix * matrix convolution.

TPUs provide direct between functional units for these operation patterns . . .
. . . reducing the number of register write and read operations.

TPU Examples

Google TPUs.

A highly specialized accelerators.

Accessible via Google Cloud.

NVIDIA Tensor Cores

Part of recent NVIDIA GPUs, starting with Volta/Turing.

FPGA (Field-Programmable Gate Array):
A chip that can be programmed to mimic a specific piece of digital hardware.

Suppose you have a design for a digital circuit.

You can fabricate an *ASIC* version . . .
. . . and wait months for delivery . . .
. . . and pay $100k for the first chip, and pennies for the second, etc.

Or you can download the design into an FPGA. . .
. . . and have your part in seconds . . .
. . . and pay a few dollars for the first chip, and for the second, etc.

ASICs v. FPGAs

ASICs used when:

- A large number of chips are needed.
- The design is not expected to change over the product lifetime.
- The highest performance is needed.
- The design is very large.

  Examples: cell phone signal processor, Bitcoin mining rig.

FPGAs used when:

- The design may change frequently or require updates.
- A part is needed quickly, perhaps as a prototype.

  Example: WiFi router.

FPGA Accelerator:
An accelerator which creates custom hardware, at the time a program is run, which should execute parts of the code more efficiently than any general-purpose device.

Using an FPGA Accelerator

Programming an FPGA accelerator can be more like hardware design.

Methods to simplify programming are still in the research stage.

They work best for specialized applications . . .
. . . rather than as an accelerator in a general-use facility.

Few high-performance computing facilities have FPGA accelerators.

Accelerators and this Course

GPU Accelerators

These will be covered because of their relative maturity and success.

Tensor Processing / Machine Learning Accelerators

These will be covered because of current interest and rapid advancement.

FPGA Accelerators

Not covered because they are very different than GPUs and many-cores . . .
. . . and because so far they are used in much more specialized settings.

Those interested in FPGA acceleration might look for high-level synthesis courses here.

GPU, CPU Raw Performance Numbers

GPU: NVIDIA K20x (2012)
SP FP, 3950 GFLOPS; DP FP, 1310 GFLOPS; Off-Chip Bandwidth 250 GB/s

GPU: NVIDIA P100 (2016)
SP FP, 4761 GFLOPS; DP FP, 2381 GFLOPS; Off-Chip Bandwidth 549 GB/s

GPU: NVIDIA V100 (2018)
SP FP, 7066 GFLOPS; DP FP, 3533 GFLOPS; Off-Chip Bandwidth 898 GB/s

Many Core: Xeon Phi 7120 P
SP FP, 2416 GFLOPS; DP FP 1208 GFLOPS, Off-Chip Bandwidth 352 GB/s

CPU: Intel Xeon E7-8870. (2011) (Ten cores, 2.4 GHz, 128b vector insn.)
SP FP, 96 GFLOPS; DP FP 48 GFLOPS, Off-Chip Bandwidth ? GB/s

CPU: Intel Xeon W-2195. (2017) (18 cores, 2.3 GHz, 512b vector insn.)
SP FP, 1325 GFLOPS; DP FP 662 GFLOPS, Off-Chip Bandwidth 85.3 GB/s

GPU v. CPU

GPU can execute floating-point operations at a higher rate . . .
. . . because it has more floating point units.

GPU can read and write memory at a higher rate.

CPUs are easier to program.

CPUs can run certain programs faster.

CPU/GPU Similarities

Both run programs.

Machine languages are similar.

Both have instructions for integer and FP operations.

High-end chips are roughly same area, and draw same power.

GPU Incidental (and diminishing) Differences

Special hardware for *texture fetch and filtering* operations.

Special hardware for *interpolation*.

Trend is less specialized hardware.

GPU Important Performance Spec Differences

GPUs can do more FP operations per second.

GPUs can transfer more data per second.

GPU Programmability Differences

Extensive tuning required to achieve performance.

GPUs perform poorly on certain problems, regardless of tuning.

GPUs succeeded where many failed: Establishing a new computer architecture.

CPUs today share similar architecture.

Very good example of a one-size-fits-all device . . .
. . . same chip in business, scientific computation, server, etc.

Exceptions are minor: embedded.

In the past specialized architectures could be successful.

Cray supercomputers.

But in most cases, different or specialized architectures failed:

Database machines.

LISP machines.

Specialized architectures failed because:

They were not that much faster.

Were very expensive to develop.

Market was small and so engineering big part of cost.

But two specialized architectures *have* succeeded: GPUs and TPUs.

GPUs: First, their birth.

3D computer graphics is compute intensive. (Think frame rate.) Need.

Graphics computation is well structured.

Facilitates development of specialized processor.

Amount of code relatively small.

Can be isolated in libraries.

Large market: Home computers, game consoles.

Can amortize development costs.

Their evolution to non-graphical use.

Computational characteristics of 3D graphics code shared by other types of programs.

GPUs could easily be modified to support non-graphical use.

References:

[1] Dally, W. J., Keckler, S. W., and Kirk, D. B. Evolution of the graphics processing unit (GPU). *IEEE Micro 41*, 06 (nov 2021), 42–51. `http://dx.doi.org/10.1109/MM.2021.3113475`.

[2] Jouppi, N. P., Yoon, D. H., Kurian, G., Li, S., Patil, N., Laudon, J., Young, C., and Patterson, D. A domain-specific supercomputer for training deep neural networks. *Commun. ACM 63*, 7 (June 2020), 6778. `https://doi.org/10.1145/3360307`.

[3] NVIDIA Corporation. NVIDIA A100 tensor core GPU architecture. Tech. rep., NVIDIA Corporation, 2020.