UNITED STATES DISTRICT COURT
DISTRICT OF MASSACHUSETTS

Singular Computing LLC,	)	
	)	Civil Action No. 1:19-cv-12551
Plaintiff,	)	
	)	
v.	)	
	)	
Google LLC,	)	
	)	
Defendant.	)	**JURY TRIAL DEMANDED**
	)	

COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff, Singular Computing LLC ("Singular"), for its complaint against Defendant,

Google LLC, ("Google"), alleges as follows:

THE PARTIES

1.   Singular is a Delaware limited liability company having its principal places of

business at 10 Regent Street, Newton, MA 02465 and The Cambridge Innovation Center, 1

Broadway, Cambridge, MA 02142.

2.   Google is a Delaware limited liability company and has regular and established

places of business in this District, including a major office complex in Cambridge,

Massachusetts with over 1,500 employees. Google may be served with process through its

registered agent, Corporation Service Company, 84 State Street, Boston, MA 02109.

JURISDICTION

3.   This is a civil action for patent infringement under the patent laws of the United

States, 35 U.S.C. § 271, et seq. This Court has subject matter jurisdiction under 28 U.S.C.

§§ 1331 and 1338(a).

4.   This Court has general personal jurisdiction over Google because Google is

engaged in substantial activity, which is not isolated, at its regular and established places of

business within this judicial district. This Court has specific jurisdiction over Google because

Google has committed acts of infringement within this judicial district giving rise to this action,

and has established more than minimum contacts within this judicial district, such that the

exercise of jurisdiction over Google in this Court would not offend traditional notions of fair

play and substantial justice.

5.   Venue is proper in this judicial district pursuant to 28 U.S.C. §§ 1391(b)-(c) and

1400(b) because Google maintains regular and established places of business and has committed

acts of patent infringement within this judicial district.

FACTUAL BACKGROUND

6.   Singular was founded by Dr. Joseph Bates to, *inter alia*, design, develop, and

produce computers having new architectures, including the patented computer architectures at

issue in this case. Dr. Bates is the President and CEO of Singular. Since 2009, Singular has

continuously operated out of the Boston area.

7.   Dr. Bates' interest in computer science dates back to at least 1969, when, as a

thirteen-year-old youth, he was admitted to Johns Hopkins University. His success at this

university sparked a pilot program for exceptionally gifted youths, which program went on to

become the widely recognized Johns Hopkins Center for Talented Youth (also known as "CTY";

see https://cty.jhu.edu) that developed the talents of over 165,000 academically advanced pre-college students, including those of Google's co-founders (Sergei Brin and Larry Page).   By age seventeen, Dr. Bates had earned a bachelors and master's degree in computer science from Johns Hopkins. He earned a computer science doctoral degree from Cornell University at age twenty-three. Dr. Bates' research and teaching interests have centered around several cutting-edge computer science topics, including the design of computer programming languages, and artificial intelligence (AI) software programs (i.e., software programs that see, hear, or understand).

8.   During his career working at the vanguard of computer science, Dr. Bates realized that although the theoretical computing power inside computers (as represented by the number of transistors inside a computer) was growing exponentially under a phenomenon popularly known as Moore's Law, the vast majority of that increase in computing power was not being made available to users. Under then existing computer architectures, even computers containing over a billion transistors were architected so as to typically perform only a handful of operations per unit of time ("period") when using CPUs. Such conventional computers typically performed only a few hundred operations per period when using GPUs.

9.   The new, novel and improved computer architectures developed by Dr. Bates, provide for the inclusion within computer processors, of processing elements designed to perform low precision and high dynamic range (LPHDR) arithmetic operations. Dr. Bates' patented architectures, allow for, *inter alia*, more efficient use of a computer's transistors and have revolutionized the way AI training and inference are accomplished.

10.   For example, a multiplication operation requested by many software programs requires on the order of a million transistors per multiplication operation when using a conventional computer architecture. Implementing Dr. Bates' LPHDR architecture on the other

hand can require a far smaller number of transistors per multiplication operation, as specified in his patents. That vast difference in the required number of transistors per multiplication operation creates the opportunity to pack into a computer having a normal number of transistors (e.g., several billion transistors, for a personal computer) a very large number of LPHDR processing elements that can each perform an operation per period. Such a large number of LPHDR processing elements can collectively perform a number of operations per period that is on the order of a hundred times larger than the number of operations per period performed by a conventional computer having the same number of transistors.   A computer utilizing Dr. Bates' novel architecture achieves this advantage—executing a far larger number of operations per period than a conventional computer—while supporting software programs that require operations to be performed on numbers having high dynamic range.

11.   Dr. Bates' architectures accomplish the foregoing even though the constituent LPHDR processing elements frequently generate, in response to requests to perform arithmetic operations, results that materially differ from the exact results of those operations. Singular LPHDR processing elements used in an AI software program, for example, can generate results in response to requested arithmetic operations that differ by at least 0.2% from their respective exact results, for at least 10% of all such requested operations, and yet still enable that AI software program to function correctly. It was not obvious and was in fact counterintuitive to those skilled in the art as of 2009 to make a computer from a very large number of LPHDR processing elements that each frequently generate such materially inexact results, knowing that such a computer was going to be used by software programs to execute numerous tasks that each required hundreds, thousands or even millions of sequential arithmetic operations that could

accumulate errors. Dr. Bates, however, conceived and made such a computer utilizing his novel and patented computer architectures.

12.   In some embodiments of Singular's patented computer architectures, LPHDR processing elements can be deployed within a computer in massively parallel configurations to further amplify their relatively higher efficiency. In still other exemplary configurations, massive numbers of these LPHDR processing elements can be deployed in conjunction with far smaller numbers of higher precision processing elements found within conventional computer architectures, to extend the range of software programs that can benefit from Singular's high-efficiency computing architecture.

13.   Singular's revolutionary approach to computer architecture is described in a provisional patent application entitled "Massively Parallel Processing with Compact Arithmetic Element" that was filed in June of 2009 and made public in June of 2010.

14.   After filing this seminal patent application, Singular under the direction of Dr. Bates built a computer incorporating its novel architecture. The Singular prototype was able to execute a software program that performed conventional neural network image classification, for example, at a rate 30 times faster than a conventional computer having comparable physical characteristics in terms of its number of transistors, its semiconductor fabrication process and power draw.

15.   As Singular was designing and building prototypes of its new computer, Google was belatedly recognizing the limitations of its conventional computer architectures in providing users with computer-based services such as Translate, Photos, Search, Assistant, and Gmail. According to Google, it was hurtling towards a "scary and daunting" situation. The situation arose as Google was starting to improve these services by running AI software

programs on its computers, and as those services consequently became more popular. According

to Google, it was "scary and daunting" because the new AI software programs being run on the

computers in its data centers required far more computer operations per period than the software

programs Google was previously executing. For example, by its own estimation, Google would

have to at least double its computing footprint just to keep up with the increased computer

requirements being driven by improved AI-based speech recognition services alone. Google

realized it needed Dr. Bates' computer architectures to solve this "daunting" situation.

16.   Google's infringement of U.S. Patents 8,407,273 and 9,218,156 is willful.

17.   Less than 2 years after the filing of the provisional application, Dr. Bates and

Google executed a non-disclosure agreement (NDA) prepared by Google.
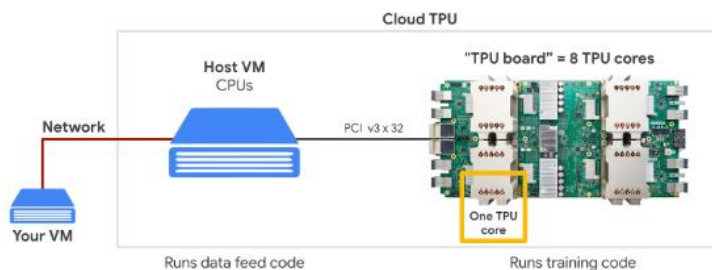
18.   After the filing of the provisional patent application, Dr. Bates met with

representatives from Google more than three times prior to early 2017.

19.   During the course of these meetings, Dr. Bates disclosed his computer

architectures and prototype. Dr. Bates also advised Google such was patent-protected.

20.   Google knew or should have known of the '273 and '156 patents prior to the

launch of the accused Cloud Tensor Processing Unit Version 2 (TPUv2 Device) in May 2017.

TPU versions

Each TPU version defines the specific hardware characteristics of a TPU device. The TPU version defines the architecture for each TPU core, the amount of high-bandwidth memory (HBM) for each TPU core, the interconnects between the cores on each TPU device, and the networking interfaces available for inter-device communication.

21.   Prior to the launch of the accused TPUv2 Device and Cloud Tensor Processing Unit Version 3 (TPUv3 Device), Google knew or should have known that the accused devices infringed the '273 and '156 patents or was willfully blind to such infringement.
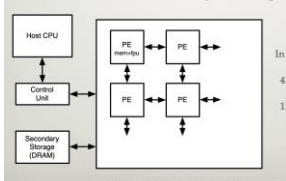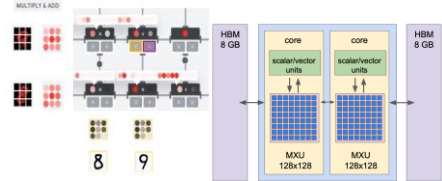
22.   Following disclosure to Google by Dr. Bates of his invention, Google copied and adopted Dr. Bates' patented invention, incorporating such into the accused TPUv2 and TPUv3 Devices and more generally into its data centers. This is apparent from a comparison of Dr. Bates' patented architecture and that of the accused TPUv2 and TPUv3 Devices. It is also apparent from an exemplary comparison of the disclosures made in writing by Dr. Bates to Google from 2010-2014 with the properties and features that Google later adopted in its TPUv2 and TPUv3 Devices in 2017-2018. For example:

Singular Presentations Made to Google / Jeff Dean (2010-2014)	**Google Documents**
	Google Publication of TPUv2 (2017) and TPUv3 (2018)

Singular Presentations Made to Google / Jeff Dean (2010-2014)	**Google Documents**
	"We started to look at what we could do for these kinds of deep learning models that could be more computationally efficient and there are two really nice properties that deep learning models have. First, they are very tolerant of reduced precision... you don't need 6 or 7 digits of precision like you would in floating point computations or even more in double computations… you can build hardware that is only designed to accelerate low precision linear algebra, you're golden, and that enables you then really tailor the hardware to do only that,"
	"Around the time of maybe 2011, 2012, when the Google Brain project that I co-founded was just getting started, we started to collaborate with . . . the speech recognition team [at Google] . . . and so we could tell that as speech recognition gets better people are going to use it more and more . . . and at the time, we had [sic] lots and lots of CPUs in our data center and if you look at how much computation that would be required if a hundred million of our users started to do that, that was actually kind of daunting and scary, we would have essentially double the computing footprint of Google just to support like a slightly better speech recognition model.

23.   Prior to 2017, Google knew that its demand for AI-based user services far exceeded its computing capabilities. Google recognized that, but for its inclusion of the technology covered by Dr. Bates' patents inside its computers, it would have had to at least double its computing footprint to accommodate such demand for delivering increased speech recognition services alone.

24.   As of 2017, Google housed its service-providing computers in the United States in at least eight data centers. As of 2017, the approximate cost to build each data center was at

least 1.25 billion dollars. As recognized by Google, but for the incorporation of the technology covered by Dr. Bates' patented invention, Google would have to at least double the number of data centers in the U.S. at a cost of at least 10 billion dollars to accommodate increased demand.

25.   Google uses the accused TPUv2 and TPUv3 Devices to provide AI capabilities that enhance the performance and efficacy of its Ads platform (e.g. determining which ads to serve to which users to maximize revenue to Google), as well as its Translate, Photos, Search, Assistant, Cloud and Gmail services. Google provides Translate, Photos, Search, Assistant, Cloud, and Gmail services to the public and leverages public engagement with these services to enhance its Ads platform. As a result, Google services generate at least tens of billions of dollars per year in profit.

26.   Google now operates at least eleven data centers in the USA. On information and belief, Google's infringing TPUv2 and TPUv3 Devices are installed at and operate in each of these data centers. These include Google's USA-based data centers at: Berkeley County, South Carolina; Council Bluffs, Iowa; The Dalles, Oregon; Douglas County, Georgia; Henderson, Nevada; Jackson County, Alabama; Lenior, North Carolina; Loudoun County, Virginia; Mayes County, Oklahoma; Midlothian, Texas; and Montgomery County, Tennessee.

THE PATENTS-IN-SUIT

27.   On March 26, 2013, the USPTO issued United States Patent No. 8,407,273 ("the '273 patent"), titled PROCESSING WITH COMPACT ARITHMETIC PROCESSING ELEMENT. On December 22, 2015, the USPTO issued United States Patent No. 9,218,156 ("the '156 patent"), titled PROCESSING WITH COMPACT ARITHMETIC PROCESSING ELEMENT. On September 17, 2019, the USPTO issued United States Patent No. 10,416,961 ("the '961 patent"), titled PROCESSING WITH COMPACT ARITHMETIC PROCESSING

ELEMENT. The '273 patent, '156 patent, and '961 patent (collectively the "patents-in-suit") are each valid and enforceable.

28.   Singular is the owner and assignee of all rights, title and interest in and to the patents-in-suit, and holds all substantial rights therein, including the right to grant licenses, to exclude others, and to enforce and recover past damages for infringement. The assignment of rights for the '156 patent was duly recorded at the USPTO on March 25, 2013. The assignment of rights for the '273 patent was duly recorded at the USPTO on February 17, 2012. The assignment of rights for the '961 patent was duly recorded at the USPTO on October 30, 2018.

29.   The application for the patents-in-suit was first filed by inventor Dr. Bates as a provisional patent application (Application No. 61/218,691) on June 19, 2009.

COUNT I
(Google's Infringement of United States Patent No. 8,407,273)

30.   Paragraphs [1-29] are reincorporated by reference as if fully set forth herein.

31.   The '273 patent addresses, *inter alia*, the aforementioned technological problem of a computer making inefficient use of its transistors.

32.   The '273 patent teaches a technological solution to this problem in the form of novel, unconventional and counterintuitive computer architectures that include, *inter alia*, the following:

(i)   at least one LPHDR execution unit (e.g., a processing element) that

   a.   accepts input signals representing numerical values, that each have a dynamic range that is at least as wide as from 1,000,000 to 1/1,000,000, and

   b.   produces output signals representing numerical values, in response to requested arithmetic operations, that differ by at least 0.05% from their

respective exact results for at least 5% of all possible valid such requested

operations; and,

(ii)   a number of LPHDR execution units that exceeds by at least 100 the number of

higher precision (e.g., 32 bit) floating point multiplication processing elements.

Dr. Bates' patented architectures solve the aforementioned problem of inefficient transistor use.

Dr. Bates' LPHDR processing elements utilize a far smaller number of transistors per operation

than the high-precision processing elements of conventional computer architectures. This

difference in the required number of transistors per operation creates the opportunity to pack into

a computer having a normal number of transistors (e.g., several billion transistors) a very large

number of LPHDR processing elements that can collectively perform a number of multiplication

operations per period that is many multiples larger than the number of multiplication operations

per period provided by a conventionally architected computer having similar physical

characteristics (i.e., in terms of its number of transistors, its semiconductor fabrication process,

power draw, etc.).   A computer utilizing Dr. Bates' novel architecture achieves the advantage of

executing a larger number of operations per period than a conventional computer while

supporting software programs that require operations to be performed on numbers having high

dynamic range.

33.   Dr. Bates' architectural solution to the aforementioned problem of inefficient

transistor usage represented a fundamentally new, unconventional and novel approach to

computer architecture. Dr. Bates' patented architectural solutions were not obvious, or

conventional to one of ordinary skill in the art at the time of the invention. Conventional

computer architectures, for example, even when intended for execution of AI software programs,

did not include the concept of a computer based on a large number of low precision high

dynamic range (LPHDR) processing elements. Prior to Dr. Bates' invention, such a computer

did not exist.

34.   Computer architects as of 2009 taught away from Dr. Bates' invention. Dr.

Bates' LPHDR processing elements frequently generate, in response to requests to perform

arithmetic operations on high dynamic range numbers, results that materially differ from the

exact results of those operations. For example, the '273 patent teaches an LPHDR processing

element that can generate results in response to requested arithmetic operations that differ by at

least 0.05% from their respective exact results, for at least 5% of all possible valid such

requested operations. It would have been counterintuitive to those in the art as of 2009, to

architect a computer from even one such frequently inexact LPHDR processing element, let

alone a large number of LPHDR processing elements that were all frequently generating such

materially inexact results for a given software program. It was counterintuitive to architect a

computer comprising numerous LPHDR processing elements that each were frequently

generating materially inexact results, knowing that such a computer was going to be used by

software programs to execute many tasks that each required hundreds, thousands or even

millions of sequential arithmetic operations that could accumulate errors. Dr. Bates nonetheless

conceived, made and patented a working computer utilizing such LPHDR processing elements.

35.   The '273 patent ushered in a revolutionary increase in computer efficiency

through improved computer architectures. Its claims specify architectural features pertaining to a

computer design, including LPHDR processing elements, and heterogeneous computers based on

particular ratios of LPHDR processing elements to conventional higher precision processing

elements.

36.   Google has directly infringed, and continues to directly infringe, literally and/or by the doctrine of equivalents, at least claims 17, 18, 52 and 53 of the '273 patent by making, using, testing, selling, offering for sale and/or importing into the United States its TPUv2 and TPUv3 Devices alone or in combination with its existing data servers. Google's computer systems that infringe the '273 patent include the TPUv2 and TPUv3 Devices. The infringing TPUv2 and TPUv3 Devices, in Google's own words, power at least Google Translate, Photos, Search, Assistant, and Gmail, as published by Google:

Empowering businesses with Google Cloud AI

Machine learning has produced business and research breakthroughs ranging from network security to medical diagnoses. We built the Tensor Processing Unit (TPU) in order to make it possible for anyone to achieve similar breakthroughs. Cloud TPU is the custom-designed machine learning ASIC that powers Google products like Translate, Photos, Search, Assistant, and Gmail. Here's how you can put the TPU and machine learning to work accelerating your company's success, especially at scale.

37.   Claim 53 of the '273 patent is reproduced below:

A device:
comprising at least one first low precision high-dynamic range
(LPHDR) execution unit adapted to execute a first operation on a first input
signal representing a first numerical value to produce a first output signal
representing a second numerical value,
wherein the dynamic range of the possible valid inputs to the first operation is at
least as wide as from 1/1,000,000 through 1,000,000 and for at least X=5% of the
possible valid inputs to the first operation, the statistical mean, over repeated
execution of the first operation on each specific input from the at least X % of the
possible valid inputs to the first operation, of the numerical values represented by
the first output signal of the LPHDR unit executing the first operation on that
input differs by at least Y=0.05% from the result of an exact mathematical
calculation of the first operation on the numerical values of that same input;
wherein the number of LPHDR execution units in the device exceeds by at least
one hundred the non-negative integer number of execution units in the device

adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide.

38.   A TPUv2 and a TPUv3 Device are examples of a "*device*," as claimed by the '273 patent. As published by Google:

TPU versions

Each TPU version defines the specific hardware characteristics of a TPU device. The TPU version defines the architecture for each TPU core, the amount of high-bandwidth memory (HBM) for each TPU core, the interconnects between the cores on each TPU device, and the networking interfaces available for inter-device communication.

Cloud TPU

When you request one "Cloud TPU v2" on Google Cloud Platform, you get a virtual machine (VM) which has a PCI-attached TPU board. The TPU board has four dual-core TPU chips. Each TPU core features a VPU (Vector Processing Unit) and a 128x128 MXU (MatriX multiply Unit). This "Cloud TPU" is then usually connected through the network to the VM that requested it. So the full picture looks like this:

39.   Each TPUv2 and TPUv3 Device infringes claim 53 of the '273 patent, by *inter alia*, including over 100,000 matrix multiplication unit (MXU) arithmetic logic units (ALUs) and associated circuitry.

  a.   Each TPUv2 Device has 8 MXUs (one MXU per TPU core, 2 TPU cores per chip, and 4 chips per TPUv2 Device), and each TPUv3 Device has 16 MXUs (two MXUs per TPU core, 2 TPU cores per chip, and 4 chips per TPUv3 Device). As published by Google:

- TPU v2:
    - 8 GiB of HBM for each TPU core
    - One MXU for each TPU core
    - Up to 512 total TPU cores and 4 TiB of total memory in a TPU Pod
- TPU v3:
    - 16 GiB of HBM for each TPU core
    - Two MXUs for each TPU core
    - Up to 2048 total TPU cores and 32 TiB of total memory in a TPU Pod

TPU v2 - 4 chips, 2 cores per chip   TPU v3 - 4 chips, 2 cores per chip
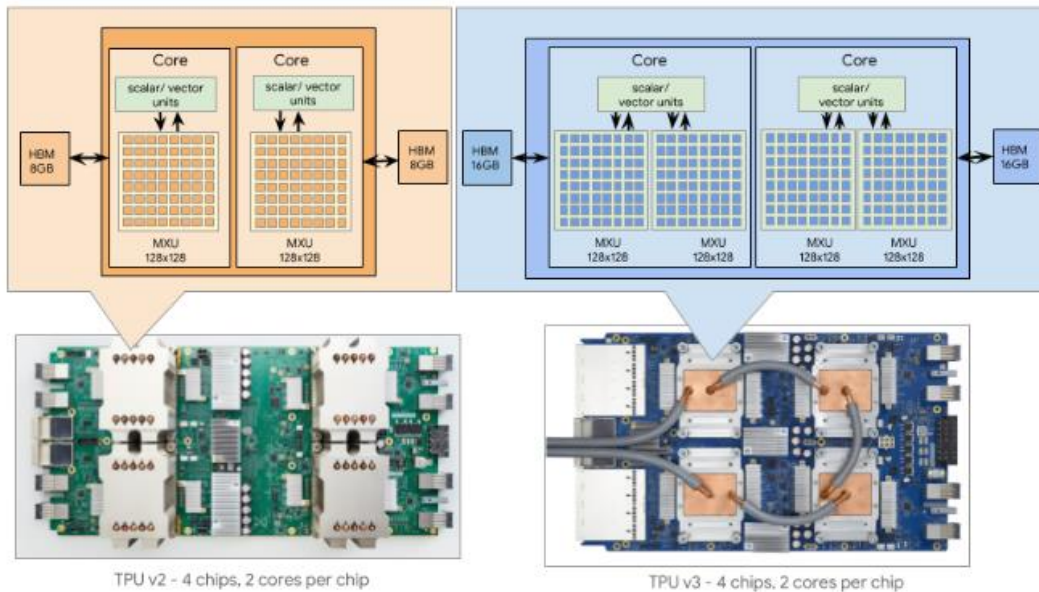
b.   Each MXU contains a systolic array having 128 X 128 MXU ALUs (i.e., 16,384 ALUs). A TPUv2 Device has 131,072 MXU ALUs and a TPUv3 Device has 262,144 MXU ALUs. As published by Google:

Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication unit (MXU),   a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then automatically converts them to bfloat16 before calculation. (A TPU can perform FP32 multiplications via multiple iterations of the MXU.) Inside the MXU, multiplications are performed in bfloat16 format, while accumulations are performed in full FP32 precision.

c.   Each of those one-hundred-thousand-plus ALUs is adapted to execute a

multiplication operation on a value that was converted to a "bfloat16" format after

being taken as input in 32-bit floating point format ("FP32 format" or "float32").

The circuitry for taking a float32 input signal, converting it to a bfloat16 value,

and then multiplying the value, is hereinafter an "MXU Reduced Precision

Multiply Cell." An MXU Reduced Precision Multiply Cell comprises the part of

an MXU ALU that performs a multiplication operation, and circuitry for taking a

float32 input signal and converting it to a bfloat16 value. As published by Google:

Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication
unit (MXU),   a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and
multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the
majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then
automatically converts them to bfloat16 before calculation. (A TPU can perform
FP32 multiplications via multiple iterations of the MXU.) Inside the MXU,
multiplications are performed in bfloat16 format, while accumulations are
performed in full FP32 precision.

Cloud TPU
System Architecture

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute
power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each
cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies
at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better
training and model accuracy than the IEEE half-precision representation.

40.   Over 100,000 individual MXU Reduced Precision Multiply Cells are in each

TPUv2 and TPUv3 Device. Each individual MXU Reduced Precision Multiply Cell is a "*low*

precision high-dynamic range (LPHDR) execution unit adapted to execute a first operation on a

first input signal representing a first numerical value to produce a first output signal

representing a second numerical value." The "*first operation*" executed by each individual

MXU Reduced Precision Multiply Cell is a multiplication operation that is (i) performed on two

input signals each representing a float32 numerical value, but (ii) carried out at "reduced

bfloat16 precision." Such an operation (e.g., "X[2,0]*W[0,0]" in the example equation for

Y[2,0] that Google provides below) is a part of a larger float32 matrix multiplication operation

(e.g., Y= X*W in the example Google provides below) being performed at "reduced bfloat16

precision" by the MXU as a whole. As published by Google:

> Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication
> unit (MXU),   a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and
> multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the
> majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then
> automatically converts them to bfloat16 before calculation. (A TPU can perform
> FP32 multiplications via multiple iterations of the MXU.) Inside the MXU,
> multiplications are performed in bfloat16 format, while accumulations are
> performed in full FP32 precision.

> **Systolic array**
>
> The MXU implements matrix multiplications in hardware using a so-called 'systolic array' architecture in which data
> elements flow through an array of hardware computation units. (In medicine, 'systolic" refers to heart contractions and
> blood flow, here to the flow of data.)
>
> The basic element of a matrix multiplication is a dot product between a line from one matrix and a column from the other
> matrix (see illustration at the top of this section). For a matrix multiplication Y=X*W, one element of the result would be:
>
> $$Y[2,0] = X[2,0]*W[0,0] + X[2,1]*W[1,0] + X[2,2]*W[2,0] + \ldots + X[2,n]*W[n,0]$$

Illustration: a dense neural network layer as a matrix multiplication, with a batch of eight images processed through the neural network at once. Please run through one line x column multiplication to verify that it is indeed doing a weighted sum of all the pixels values of an image. Convolutional layers can be represented as matrix multiplications.

Cloud TPU

System Architecture

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better training and model accuracy than the IEEE half-precision representation.

41.   The "*first input signal*" for each individual MXU Reduced Precision Multiply Cell is the signal representing a float32 value from inside either matrix that is being multiplied by the MXU. In the illustration below taken from a Google animation, such input signals are depicted using the red dots moving rightwards into the left of the MXU and its Reduced Precision Multiply Cells. Such input signals are also depicted in the same Google animation using the black/grey dots moving upwards into the bottom of the MXU and its Reduced Precision Multiply Cells. The "*first output signal*" produced by an MXU Reduced Precision Multiply Cell is the result of the float32 multiplication operation it performs at reduced bfloat16

18

precision. Such a *"first output signal"* flows from the multiplier "x" to the adjacent adder "+" in one of the Google illustrations below and is described as a result. In Google's own words, "as each multiplication is executed, the result will be passed to the next multipliers while taking summation at the same time." As published by Google:

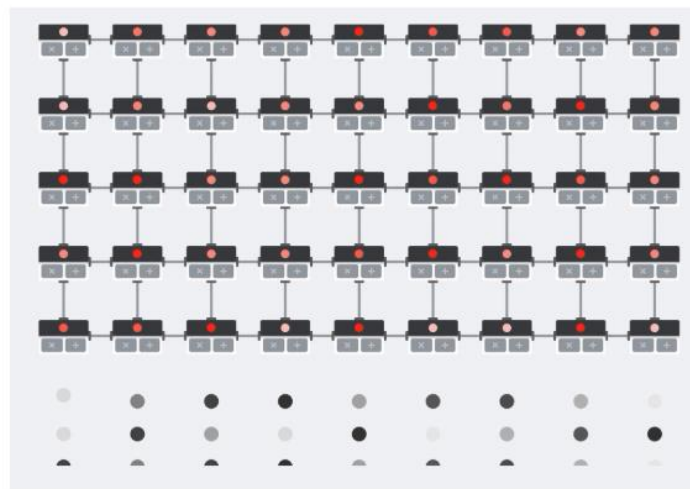Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication unit (MXU),   a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then automatically converts them to bfloat16 before calculation. (A TPU can perform FP32 multiplications via multiple iterations of the MXU.) Inside the MXU, multiplications are performed in bfloat16 format, while accumulations are performed in full FP32 precision.

Cloud TPU
System Architecture

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better training and model accuracy than the IEEE half-precision representation.

On a GPU, one would program this dot product into a GPU "core" and then execute it on as many "cores" as are available in parallel to try and compute every value of the resulting matrix at once. If the resulting matrix is 128x128 large, that would require 128x128=16K "cores" to be available which is typically not possible. The largest GPUs have around 4000 cores. A TPU on the other hand uses the bare minimum of hardware for the compute units in the MXU: just `bfloat16 x bfloat16 => float32` multiply-accumulators, nothing else. These are so small that a TPU can implement 16K of them in a 128x128 MXU and process this matrix multiplication in one go.

Illustration: the MXU systolic array. The compute elements are multiply-accumulators. The values of one matrix are loaded into the array (red dots). Values of the other matrix flow through the array (grey dots). Vertical lines propagate the values up. Horizontal lines propagate partial sums. It is left as an exercise to the user to verify that as the data flows through the array, you get the result of the matrix multiplication coming out of the right side.

Let's see how a systolic array executes the neural network calculations. At first, the TPU loads the parameters from memory into the matrix of multipliers and adders.

Then, the TPU loads data from memory. As each multiplication is executed, the result will be passed to the next multipliers while taking the summation at the same time. So the output will be the summation of all multiplication results between data and parameters. During the whole process of massive calculations and data passing, no memory access is required at all.

42.   The "*first numerical value*" represented by the "*first input signal*," and the

"*second numerical value*" represented by the "*first output signal*," are all float32 numbers.

Float32 numbers are "*numerical values*." As published by Google:

> Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication unit (MXU),   a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then automatically converts them to bfloat16 before calculation. (A TPU can perform FP32 multiplications via multiple iterations of the MXU.) Inside the MXU, multiplications are performed in bfloat16 format, while accumulations are performed in full FP32 precision.
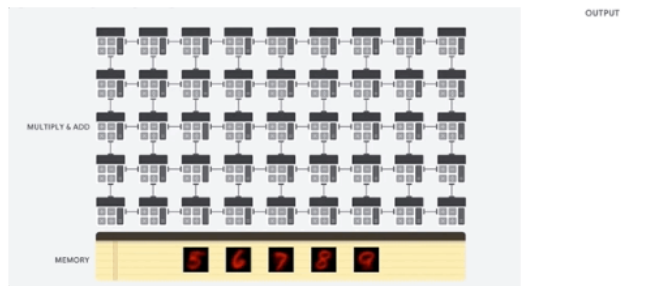
TPU v2

Cloud TPU

System Architecture

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better training and model accuracy than the IEEE half-precision representation.

Let's see how a systolic array executes the neural network calculations. At first, the TPU loads the parameters from memory into the matrix of multipliers and adders.
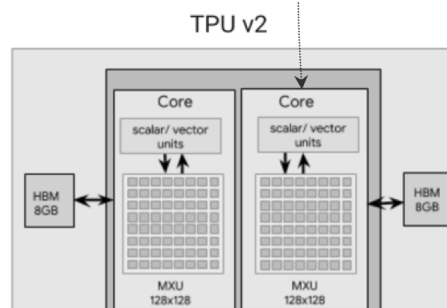

Then, the TPU loads data from memory. As each multiplication is executed, the result will be passed to the next multipliers while taking the summation at the same time. So the output will be the summation of all multiplication results between data and parameters. During the whole process of massive calculations and data passing, no memory access is required at all.

Single-precision floating-point format

From Wikipedia, the free encyclopedia

Single-precision floating-point format is a computer number format, usually occupying 32 bits in computer memory; it represents a wide dynamic range of numeric values by using a floating radix point.

A floating-point variable can represent a wider range of numbers than a fixed-point variable of the same bit width at

43.   Each MXU Reduced Precision Multiply Cell is an "*LPHDR execution unit*."

Specifically:

- For each MXU Reduced Precision Multiply Cell, "*the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/1,000,000 through 1,000,000.*" As shown above, each MXU Reduced Precision Multiply Cell performs a

float32 multiplication operation at "reduced bfloat16 precision" on valid input signals representing numerical values having a float32 format. A float32 numerical value, whose format is shown below, has the following dynamic range:

Minimum: $2^{-126} \approx 1.175494351 \times 10^{-38}$

Maximum: $(2 - 2^{-23}) \times 2^{127} \approx 3.402823466 \times 10^{38}$

As published by Google:

(a) fp32: Single-precision IEEE Floating Point Format   Range: ~1e⁻³⁸ to ~3e³⁸

(b) fp16: Half-precision IEEE Floating Point Format   Range: ~5.96e⁻⁸ to 65504

(c) bfloat16: Brain Floating Point Format   Range: ~1e⁻³⁸ to ~3e³⁸

As Figure 1 shows, bfloat16 has a greater dynamic range—i.e., number of exponent bits—than FP16. In fact, the dynamic range of bfloat16 is identical to that of FP32. We've trained a wide range of deep learning models, and in our experience, the bfloat16 format works as well as the FP32 format while delivering increased performance and reducing memory usage.

- For each MXU Reduced Precision Multiply Cell, "*for at least X=5% of the possible valid inputs to the first operation… the numerical values represented by the first output signal of the LPHDR unit executing the first operation on that input differs by at least Y=0.05% from the result of an exact mathematical calculation of the first operation on the numerical values of that same input*." Specifically, each TPUv2 and TPUv3 MXU Reduced Precision Multiply Cell performs a float32 multiplication operation but does so in Google's own words at "reduced bfloat16 precision." Each MXU Reduced Precision Multiply Cell takes the following steps: (i) receives as input two signals that each represent a float32 numerical value, (ii) converts each of the received float32 numerical

values to a bfloat16 numerical value, (iii) multiplies the resulting pair of bfloat16

numerical values with each other, and (iv) adjusts the format of the result of the bfloat16

multiplication generated in step (iii), if needed, to produce an output signal that

represents a float32 numerical value to be accumulated. As published by Google:

Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication
unit (MXU),   a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and
multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the
majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then
automatically converts them to bfloat16 before calculation. (A TPU can perform
FP32 multiplications via multiple iterations of the MXU.) Inside the MXU,
multiplications are performed in bfloat16 format, while accumulations are
performed in full FP32 precision.

Cloud TPU

System Architecture

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute
power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each
cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies
at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better
training and model accuracy than the IEEE half-precision representation.

When the float32 numerical values produced by the TPU's float32 multiplication

operation (which, as shown above, is performed at "reduced bfloat16 precision"), for a

mathematically representative sample of all possible valid pairs of inputted float32

numerical values, are compared to the numerical values produced by the exact full

precision multiplication operations for those same respective valid pairs of inputted

float32 numerical values, the TPU's float32 numerical values differ, for at least 5% of

those multiplied pairs, from the respective exact full precision values, by at least 0.05%.

This is illustrated by the Singular test results shown below.

		bf16
% of valid > 1.00%		4.65%
% of valid > 0.50%		55.39%
% of valid > 0.20%		92.69%
% of valid > 0.10%		98.15%
% of valid > 0.05%		99.52%

- For each MXU Reduced Precision Multiply Cell, *"the statistical mean, over repeated execution of the first operation on each specific input from the at least X % of the possible valid inputs to the first operation, of the numerical values represented by the first output signal of the LPHDR unit executing the first operation on that input,"* will simply equal the numerical value represented by the output signal produced when the MXU Reduced Precision Multiply Cell (i.e., an LPHDR unit) executes an operation on input signals. Each MXU Reduced Precision Multiply Cell is part of a TPUv2 Device or a TPUv3 Device, which are deterministic in their designs (i.e., an operation repeatedly performed by a TPUv2 or a TPUv3 Device on a given set of inputs signals will always yield the same output signal). As published by Google:

> Because general-purpose processors such as CPUs and GPUs must provide good performance across a wide range of applications, they have evolved myriad sophisticated, performance-oriented mechanisms. As a side effect, the behavior of those processors can be difficult to predict, which makes it hard to guarantee a certain latency limit on neural network inference. In contrast, TPU design is strictly minimal and deterministic as it has to run only one task at a time: neural network prediction. You can see its simplicity in the floor plan of the TPU die.

44.   In each TPUv2 and each TPUv3 Device, *"the number of LPHDR execution units in the device exceeds by at least one hundred the non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide."* As shown above, a TPUv2 Device has 8 MXUs, a

TPUv3 has 16 MXUs, and each MXU has 16,384 MXU Reduced Precision Multiply Cells. As also shown above, each MXU Reduced Precision Multiply Cell is an LPHDR execution unit that performs a float32 multiplication operation at "reduced bfloat16 precision." Therefore, each TPUv2 Device has at least 131,072 MXU Reduced Precision Multiply Cells that are each LPHDR execution units, and each TPUv3 has at least 262,144 MXU Reduced Precision Multiply Cells that are each LPHDR execution units. By contrast, there are far fewer execution units adapted to execute the operation of multiplication on floating point numbers that are at least 32 bits wide, on TPUv2 or TPUv3 Devices, since:

- the MXU delivers the "bulk" of a TPU Device's overall arithmetic computation capability. As published by Google:

Cloud TPU
System Architecture

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better training and model accuracy than the IEEE half-precision representation.

- the MXU Reduced Precision Multiply Cells, in performing float32 multiplication operations at "reduced bfloat16 precision," do not "*execute the operation of multiplication on floating point numbers that are at least 32 bits wide.*" Arithmetic (including multiplication) on floating point numbers that are at least 32 bits wide is defined in Singular's patents as "high dynamic range arithmetic of traditional precision." The float32 multiplication operation of an MXU Reduced Precision Multiply Cell is performed at "reduced bfloat16 precision," which entails performing multiplication on floating point numbers that are 16 bits wide. Therefore, the MXU Reduced Precision Multiply Cells themselves do not "*execute the operation of multiplication on floating point numbers that are at least 32 bits wide.*" As published by Google:

25

around 4000 cores. A TPU on the other hand uses the bare minimum of hardware for the compute units in the MXU: just `bfloat16 x bfloat16 => float32` multiply-accumulators, nothing else. These are so small that a TPU can implement 16K of them in a 128x128 MXU and process this matrix multiplication in one go.

> Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication unit (MXU), a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then automatically converts them to bfloat16 before calculation. (A TPU can perform FP32 multiplications via multiple iterations of the MXU.) Inside the MXU, multiplications are performed in bfloat16 format, while accumulations are performed in full FP32 precision.

Cloud TPU
System Architecture

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better training and model accuracy than the IEEE half-precision representation.

As affirmed by Dr. Patterson of Google at a lecture presented at the MIT Computer Science and Artificial Intelligence Lab on October 16, 2019, there is roughly a 10:1 ratio of multipliers in MXU versus VPU. The number of MXU Reduced Precision Multiply Cells (i.e., LPHDR execution units, as shown above) in each TPUv2 Device and TPUv3 Device (i.e., 131,072 for a TPUv2 Device or 262,144 for a TPUv3 Device) far exceeds the number of execution units in those same TPUv2 and TPUv3 Devices adapted to execute multiplication on floating point numbers that are at least 32 bits wide (roughly 13,107 for a TPUv2 Device, and roughly 26,214 for a TPUv3 Device). As published by Google:

MXU and VPU

A TPU v2 core is made of a Matrix Multiply Unit (MXU) which runs matrix multiplications and a Vector Processing Unit (VPU) for all other tasks such as activations, softmax, etc. The VPU handles float32 and int32 computations. The MXU on the other hand operates in a mixed precision 16-32 bit floating point format.

TPUv2 Block Diagram

- A 128x128 systolic **Matrix Multiply Unit** (**MXU**) performs Nx128x128 matrix multiplications (peak: 32K ops/clock)
- **Transpose Reduction Permute Unit** (**TRP**) on 128x128 matricies

- **Vector Processing Unit (VPU)** 32 2D Vector registers **Vregs** + 2D Vector memory **Vmem** (16MiB)
  - ≈1/10th performance MXU
- **Core Sequencer** fetches instructions from Instruction Memory **Imem**
- **Inter-Core Interconnect (ICI)** sends messages between TensorCores
- **High Bandwidth Memory (HBM)** interposer with 2 HBM stacks / TC
  - 32 64-bit busses (20x TPUv1)
- Connects to host CPU via PCIe Gen3 x16 bus using **Host DMA Queues**

45.   In knowingly adopting Dr. Bates' patented computer architectures, Google reaps the very same benefits that were predicted by Dr. Bates in his patent application more than 10 years ago. As published by Google and predicted by Dr. Bates in his patent application:

Choosing bfloat16

Our hardware teams chose bfloat16 for Cloud TPUs to improve hardware efficiency while maintaining the ability to train accurate deep learning models, all with minimal switching costs from FP32. The physical size of a hardware multiplier scales with the *square* of the mantissa width. With fewer mantissa bits than FP16, the bfloat16 multipliers are about half the size in silicon of a typical FP16 multiplier, and they are *eight times* smaller than an FP32 multiplier!

> PEs implemented according to certain embodiments of the present invention may be relatively small for PEs that can do arithmetic. This means that there are many PEs per unit of resource (e.g., transistor, area, volume), which in turn means that there is a large amount of arithmetic computational power per unit of resource. This enables larger problems to be solved with a given amount of resource than does traditional computer designs. For instance, a digital embodiment of the present invention built as a large silicon chip fabricated with current state of the art technology might perform tens of thousand of arithmetic operations per cycle, as opposed to hundreds in a conventional GPU or a handful in a conventional multicore CPU. These ratios reflect an architectural advantage of embodiments of the present invention that should persist as fabrication technology continues to improve, even as we reach nanotechnology or other implementations for digital and analog computing.

46.   As a result of Google's infringement of the '273 patent, Singular has suffered damages in an amount to be determined at trial.

COUNT II
(Google's Infringement of United States Patent No. 9,218,156)

47.   Paragraphs [1-46] are reincorporated by reference as if fully set forth herein.

48.   The '156 patent addresses, *inter alia*, the aforementioned technological problem

of a computer making inefficient use of its transistors.

49.   The '156 patent teaches a technological solution to this problem in the form of

novel, unconventional and counterintuitive computer architectures that include, *inter alia*, the

following:

(i)   At least one LPHDR execution unit (e.g., a processing element) that:

    a.   accepts input signals representing numerical values, that each have a dynamic

       range that is at least as wide as from 1,000,000 to 1/1,000,000, and

    b.   produces output signals representing numerical values, in response to

       requested arithmetic operations, which differ by at least 0.05% different from

       their respective exact results for at least 5% of all possible valid such

       requested operations;

(ii)   a computing device adapted to control the operation of the one or more LPHDR

    execution units; and,

(iii)   a number of LPHDR execution units that exceeds by at least 100 the number of

    higher precision (e.g., 32 bit) floating point multiplication processing elements.

Dr. Bates' patented architectures solve the aforementioned problem of inefficient transistor use.

Dr. Bates' LPHDR processing elements utilize a far smaller number of transistors per operation

than the high-precision processing elements of conventional computer architectures. This

difference in the required number of transistors per operation creates the opportunity to pack into

a computer having a normal number of transistors (e.g., several billion transistors) a very large

number of LPHDR processing elements that can collectively perform a number of multiplication

operations per period that is many multiples larger than the number of multiplication operations

per period provided by a conventionally architected computer having similar physical

characteristics (i.e., in terms of its number of transistors, its semiconductor fabrication process,

power draw, etc.).   A computer utilizing Dr. Bates' novel architecture achieves the advantage of

executing a far larger number of operations per period than a conventional computer while

supporting software programs that require operations to be performed on numbers having high

dynamic range.

50.   Dr. Bates' architectural solution to the aforementioned problem of inefficient

transistor usage represented a fundamentally new and unconventional approach to computer

architecture. Dr. Bates' patented architecture was not obvious, or conventional to one of

ordinary skill in the art at the time of the invention. Conventional computer architectures, for

example, even when intended for execution of AI software programs, did not include the concept

of a computer based on low precision high dynamic range (LPHDR) processing elements.

Computer architects even taught away from this concept. Prior to Dr. Bates' invention, such a

computer did not exist.

51.   Computer architects as of 2009 taught away from Dr. Bates' invention. Dr.

Bates' LPHDR processing elements frequently generate, in response to requests to perform

arithmetic operations on high dynamic range numbers, results that materially differ from the

exact results of those operations. For example, the '156 patent teaches an LPHDR processing

element that can generate results in response to requested arithmetic operations that differ by at

least 0.05% from their respective exact results, for at least 5% of all possible valid such

requested operations. It would have been counterintuitive to those in the art as of 2009, to

architect a computer from even one such frequently inexact LPHDR processing element, let alone a large number of LPHDR processing elements that were all frequently generating such materially inexact results for a given software program. It was counterintuitive to architect a computer comprising numerous LPHDR processing elements that each frequently generate inexact results, knowing that such a computer was going to be used by software programs to execute many tasks that each required hundreds, thousands or even millions of sequential arithmetic operations that could accumulate errors. Dr. Bates nonetheless conceived, made and patented a working computer utilizing such LPHDR processing elements.

52.   The '156 patent ushered in a revolutionary increase in computer efficiency through improved computer architectures. Its claims specify architectural features pertaining to a computer design, including LPHDR processing elements, a computing device that is adapted to control the operation of the LPHDR execution units, and heterogeneous computers based on particular ratios of LPHDR processing elements to conventional higher precision processing elements.

53.   Google has directly infringed, and continues to directly infringe, literally and/or by the doctrine of equivalents, at least claims 6, 7, 21 and 22 of the '156 patent by making, using, testing, selling, offering for sale and/or importing into the United States its TPUv2 and TPUv3 Devices alone or in combination with its existing data servers. Google's computer systems that infringe the '156 patent include the TPUv2 and TPUv3 Devices. The infringing TPUv2 and TPUv3 Devices, in Google's own words, "powers" at least Google Translate, Photos, Search, Assistant, and Gmail. As published by Google:

Empowering businesses with Google Cloud AI

Machine learning has produced business and research breakthroughs ranging from network security to medical diagnoses. We built the Tensor Processing Unit (TPU) in order to make it possible for anyone to achieve similar breakthroughs. Cloud TPU is the custom-designed machine learning ASIC that powers Google products like Translate, Photos, Search, Assistant, and Gmail. Here's how you can put the TPU and machine learning to work accelerating your company's success, especially at scale.

54.   Claim 7 of the '156 patent is reproduced below:

A device comprising:
at least one first low precision high-dynamic range (LPHDR) execution unit adapted to execute a first operation on a first input signal representing a first numerical value to produce a first output signal representing a second numerical value,
wherein the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/1,000,000 through 1,000,000 and for at least X=5% of the possible valid inputs to the first operation, the statistical mean, over repeated execution of the first operation on each specific input from the at least X % of the possible valid inputs to the first operation, of the numerical values represented by the first output signal of the LPHDR unit executing the first operation on that input differs by at least Y=0.05% from the result of an exact mathematical calculation of the first operation on the numerical values of that same input; and
at least one first computing device adapted to control the operation of the at least one first LPHDR execution unit
wherein the at least one first computing device comprises at least one of a central processing unit (CPU), a graphics processing unit (GPU), a field programmable gate array (FPGA), a microcode-based processor, a hardware sequencer, and a state machine; and,
wherein the number of LPHDR execution units in the device exceeds by at least one hundred the non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide.
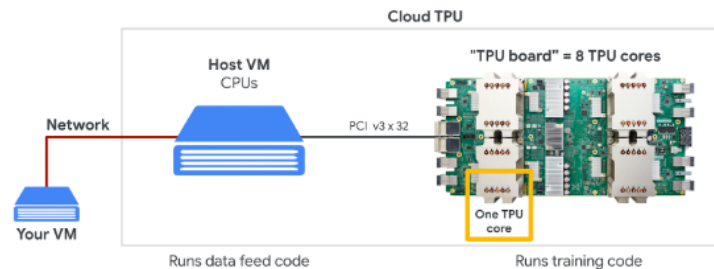
55.   A TPUv2 or a TPUv3 Device are examples of a "*device*," as claimed by the '156

patent. As published by Google:

TPU versions

Each TPU version defines the specific hardware characteristics of a TPU device. The TPU version defines the architecture for each TPU core, the amount of high-bandwidth memory (HBM) for each TPU core, the interconnects between the cores on each TPU device, and the networking interfaces available for inter-device communication.

Cloud TPU

When you request one "Cloud TPU v2" on Google Cloud Platform, you get a virtual machine (VM) which has a PCI-attached TPU board. The TPU board has four dual-core TPU chips. Each TPU core features a VPU (Vector Processing Unit) and a 128x128 MXU (MatriX multiply Unit). This "Cloud TPU" is then usually connected through the network to the VM that requested it. So the full picture looks like this:

Illustration: your VM with a network-attached "Cloud TPU" accelerator. "The Cloud TPU" itself is made of a VM with a PCI-attached TPU board with four dual-core TPU chips on it.

56.   Each TPUv2 Device and TPUv3 Device infringes claim 7 of the '156 patent by, *inter alia*, including over 100,000 matrix multiplication unit (MXU) ALUs and associated circuitry.

a.   Each TPUv2 Device has 8 MXUs (one MXU per TPU core, 2 TPU cores per chip, and 4 chips per TPUv2 Device), and each TPUv3 Device has 16 MXUs (two MXUs per TPU core, 2 TPU cores per chip, and 4 chips per TPUv3 Device). As published by Google:

- TPU v2:
    - 8 GiB of HBM for each TPU core
    - One MXU for each TPU core
    - Up to 512 total TPU cores and 4 TiB of total memory in a TPU Pod
- TPU v3:
    - 16 GiB of HBM for each TPU core
    - Two MXUs for each TPU core
    - Up to 2048 total TPU cores and 32 TiB of total memory in a TPU Pod

TPU v2 - 4 chips, 2 cores per chip   TPU v3 - 4 chips, 2 cores per chip

b.   Each MXU contains a systolic array having 128 X 128 MXU ALUs (i.e., 16,384

ALUs). Therefore, a TPUv2 Device has 131,072 MXU ALUs and a TPUv3

Device has 262,144 MXU ALUs. As published by Google:

> Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication
> unit (MXU),   a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and
> multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the
> majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then
> automatically converts them to bfloat16 before calculation. (A TPU can perform
> FP32 multiplications via multiple iterations of the MXU.) Inside the MXU,
> multiplications are performed in bfloat16 format, while accumulations are
> performed in full FP32 precision.

c.   Each of those one-hundred-thousand-plus ALUs is adapted to execute a

multiplication operation on a value that was converted to a "bfloat16" format after

33

being taken as input in 32-bit floating point format ("FP32 format" or "float32").

The circuitry for taking a float32 input signal, converting it to a bfloat16 value,

and then multiplying the value, is hereinafter an "MXU Reduced Precision

Multiply Cell." An MXU Reduced Precision Multiply Cell comprises the part of

an MXU ALU that performs a multiplication operation, and circuitry for taking a

float32 input signal and converting it to a bfloat16 value. As published by Google:

Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication
unit (MXU),   a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and
multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the
majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then
automatically converts them to bfloat16 before calculation. (A TPU can perform
FP32 multiplications via multiple iterations of the MXU.) Inside the MXU,
multiplications are performed in bfloat16 format, while accumulations are
performed in full FP32 precision.

Cloud TPU

System Architecture

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute
power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each
cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies
at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better
training and model accuracy than the IEEE half-precision representation.

57.   Over 100,000 individual MXU Reduced Precision Multiply Cells are in each

TPUv2 and TPUv3 Device. Each individual MXU Reduced Precision Multiply Cell is a "*low

precision high-dynamic range (LPHDR) execution unit adapted to execute a first operation on a

first input signal representing a first numerical value to produce a first output signal

representing a second numerical value.*" The "*first operation*" executed by each individual

MXU Reduced Precision Multiply Cell is a multiplication operation that is (i) performed on two

input signals each representing a float32 numerical value, but (ii) carried out at "reduced

bfloat16 precision." Such an operation (e.g., "X[2,0]*W[0,0]" in the example equation for

Y[2,0] that Google provides below) is a part of a larger float32 matrix multiplication operation

(e.g., Y= X*W in the example Google provides below) being performed at "reduced bfloat16 precision" by the MXU as a whole. As published by Google:

Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication unit (MXU), a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then automatically converts them to bfloat16 before calculation. (A TPU can perform FP32 multiplications via multiple iterations of the MXU.) Inside the MXU, multiplications are performed in bfloat16 format, while accumulations are performed in full FP32 precision.

Systolic array

The MXU implements matrix multiplications in hardware using a so-called "systolic array" architecture in which data elements flow through an array of hardware computation units. (In medicine, "systolic" refers to heart contractions and blood flow, here to the flow of data.)

The basic element of a matrix multiplication is a dot product between a line from one matrix and a column from the other matrix (see illustration at the top of this section). For a matrix multiplication Y=X*W, one element of the result would be:

```
Y[2,0] = X[2,0]*W[0,0] + X[2,1]*W[1,0] + X[2,2]*W[2,0] + ... + X[2,n]*W[n,0]
```
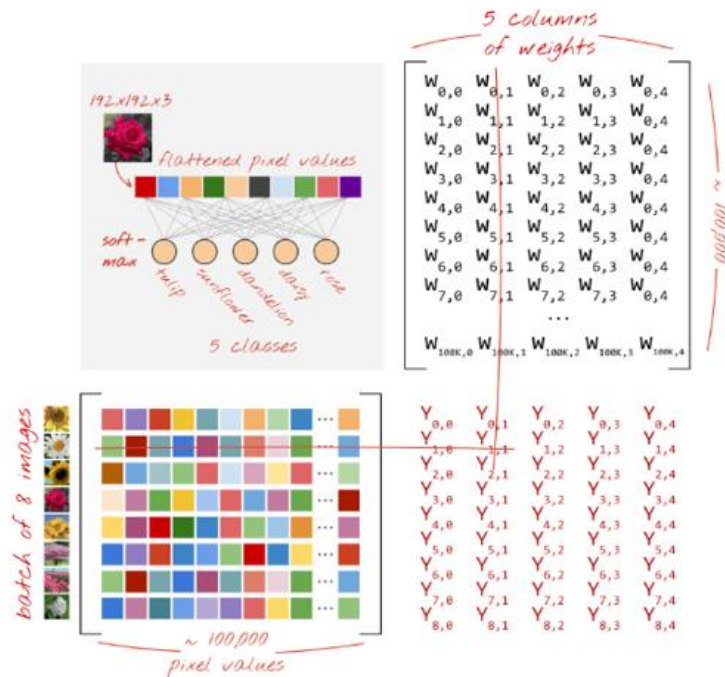

Illustration: a dense neural network layer as a matrix multiplication, with a batch of eight images processed through the neural network at once. Please run through one line x column multiplication to verify that it is indeed doing a weighted sum of all the pixels values of an image. Convolutional layers can be represented as matrix multiplications.

35

Cloud TPU

System Architecture

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better training and model accuracy than the IEEE half-precision representation.

58.   The "*first input signal*" for each individual MXU Reduced Precision Multiply

Cell is the signal representing a float32 value from inside either matrix that is being multiplied

by the MXU. In the illustration below taken from a Google animation, such input signals are

depicted using the red dots moving rightwards into the left of the MXU and its Reduced

Precision Multiply Cells. Such input signals are also depicted in the same Google animation

using the black/grey dots moving upwards into the bottom of the MXU and its Reduced

Precision Multiply Cells. The "*first output signal*" produced by an MXU Reduced Precision

Multiply Cell is the result of the float32 multiplication operation it performs at reduced bfloat16

precision. Such a *"first output signal"* flows from the multiplier "x" to the adjacent adder "+" in

the illustration below, and is described as a "result" in Google's own words: "As each

multiplication is executed, the result will be passed to the next multipliers while taking

summation at the same time." As published by Google:

Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication unit (MXU),   a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then automatically converts them to bfloat16 before calculation. (A TPU can perform FP32 multiplications via multiple iterations of the MXU.) Inside the MXU, multiplications are performed in bfloat16 format, while accumulations are performed in full FP32 precision.

Cloud TPU

System Architecture

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better training and model accuracy than the IEEE half-precision representation.

On a GPU, one would program this dot product into a GPU "core" and then execute it on as many "cores" as are available in parallel to try and compute every value of the resulting matrix at once. If the resulting matrix is 128x128 large, that would require 128x128=16K "cores" to be available which is typically not possible. The largest GPUs have around 4000 cores. A TPU on the other hand uses the bare minimum of hardware for the compute units in the MXU: just `bfloat16 x bfloat16 => float32` multiply-accumulators, nothing else. These are so small that a TPU can implement 16K of them in a 128x128 MXU and process this matrix multiplication in one go.
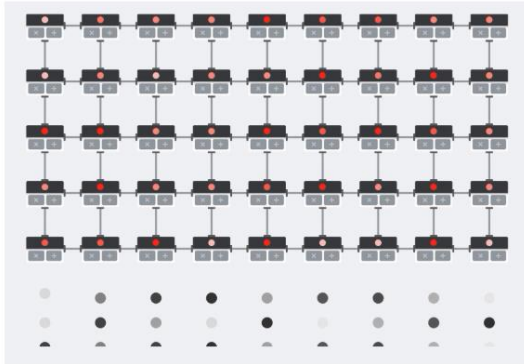

Illustration: the MXU systolic array. The compute elements are multiply-accumulators. The values of one matrix are loaded into the array (red dots). Values of the other matrix flow through the array (grey dots). Vertical lines propagate the values up. Horizontal lines propagate partial sums. It is left as an exercise to the user to verify that as the data flows through the array, you get the result of the matrix multiplication coming out of the right side.

Let's see how a systolic array executes the neural network calculations. At first, the TPU loads the parameters from memory into the matrix of multipliers and adders.

Then, the TPU loads data from memory. As each multiplication is executed, the result will be passed to the next multipliers while taking the summation at the same time. So the output will be the summation of all multiplication results between data and parameters. During the whole process of massive calculations and data passing, no memory access is required at all.

59.   The "*first numerical value*" represented by the "*first input signal*," and the

"*second numerical value*" represented by the "*first output signal*," are all float32 numbers.

Float32 numbers are "*numerical values.*" As published by Google:

Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication unit (MXU), a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then automatically converts them to bfloat16 before calculation. (A TPU can perform FP32 multiplications via multiple iterations of the MXU.) Inside the MXU, multiplications are performed in bfloat16 format, while accumulations are performed in full FP32 precision.

Cloud TPU
System Architecture

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better training and model accuracy than the IEEE half-precision representation.

Let's see how a systolic array executes the neural network calculations. At first, the TPU loads the parameters from memory into the matrix of multipliers and adders.
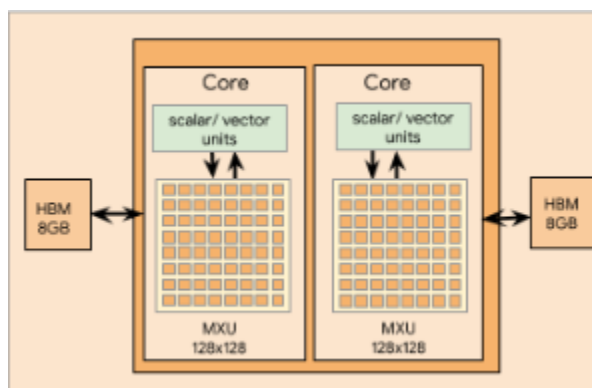

Then, the TPU loads data from memory. As each multiplication is executed, the result will be passed to the next multipliers while taking the summation at the same time. So the output will be the summation of all multiplication results between data and parameters. During the whole process of massive calculations and data passing, no memory access is required at all.

Single-precision floating-point format

From Wikipedia, the free encyclopedia

Single-precision floating-point format is a computer number format, usually occupying 32 bits in computer memory; it represents a wide dynamic range of numeric values by using a floating radix point.

A floating-point variable can represent a wider range of numbers than a fixed-point variable of the same bit width at

60.   Each MXU Reduced Precision Multiply Cell is an "*LPHDR execution unit.*"
Specifically:

- For each MXU Reduced Precision Multiply Cell, "*the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/1,000,000 through 1,000,000.*" As shown above, each MXU Reduced Precision Multiply Cell performs a float32 multiplication operation at "reduced bfloat16 precision" on valid input signals representing numerical values having a float32 format. A float32 numerical value, whose format is shown below, has the following dynamic range:

  Minimum: $2^{-126} \approx 1.175494351 \times 10^{-38}$

  Maximum: $(2 - 2^{-23}) \times 2^{127} \approx 3.402823466 \times 10^{38}$

  As published by Google:

As Figure 1 shows, bfloat16 has a greater dynamic range—i.e., number of exponent bits—than FP16. In fact, the dynamic range of bfloat16 is identical to that of FP32. We've trained a wide range of deep learning models, and in our experience, the bfloat16 format works as well as the FP32 format while delivering increased performance and reducing memory usage.

- For each MXU Reduced Precision Multiply Cell, "*for at least X=5% of the possible valid inputs to the first operation… the numerical values represented by the first output signal of the LPHDR unit executing the first operation on that input differs by at least Y=0.05% from the result of an exact mathematical calculation of the first operation on the numerical values of that same input*." Specifically, each TPUv2 and TPUv3 MXU Reduced Precision Multiply Cell performs a float32 multiplication operation but does so in Google's own words at "reduced bfloat16 precision." Each MXU Reduced Precision Multiply Cell takes the following steps: (i) receives as input two signals that each represent a float32 numerical value, (ii) converts each of the received float32 numerical values to a bfloat16 numerical value, (iii) multiplies the resulting pair of bfloat16 numerical values with each other, and (iv) adjusts the format of the result of the bfloat16 multiplication generated in step (iii), if needed, to produce an output signal that represents a float32 numerical value to be accumulated. As published by Google:

Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication unit (MXU),   a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then automatically converts them to bfloat16 before calculation. (A TPU can perform FP32 multiplications via multiple iterations of the MXU.) Inside the MXU, multiplications are performed in bfloat16 format, while accumulations are performed in full FP32 precision.

Cloud TPU

System Architecture

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better training and model accuracy than the IEEE half-precision representation.

When the float32 numerical values produced by the TPU's float32 multiplication operation (which, as shown above, is performed at "reduced bfloat16 precision"), for a mathematically representative sample of all possible valid pairs of inputted float32

numerical values, are compared to the numerical values produced by the exact full

precision multiplication operations for those same respective valid pairs of inputted

float32 numerical values, the TPU's float32 numerical values differ, for at least 5% of

those multiplied pairs, from the respective exact full precision values, by at least 0.05%.

This is illustrated by the Singular test results shown below.

		bf16
% of valid > 1.00%		4.65%
% of valid > 0.50%		55.39%
% of valid > 0.20%		92.69%
% of valid > 0.10%		98.15%
% of valid > 0.05%		99.52%

- For each MXU Reduced Precision Multiply Cell, *"the statistical mean, over repeated*

  execution of the first operation on each specific input from the at least X % of the

  possible valid inputs to the first operation, of the numerical values represented by the

  first output signal of the LPHDR unit executing the first operation on that input," will

  simply equal the numerical value represented by the output signal produced when the

  MXU Reduced Precision Multiply Cell (i.e., an LPHDR unit) executes an operation on

  input signals. Each MXU Reduced Precision Multiply Cell is part of a TPUv2 Device or

  a TPUv3 Device, which are deterministic in their designs (i.e., an operation repeatedly

  performed by a TPUv2 or a TPUv3 Device on a given set of inputs signals will always

  yield the same output signal). As published by Google:

Because general-purpose processors such as CPUs and GPUs must provide good performance across a wide range of applications, they have evolved myriad sophisticated, performance-oriented mechanisms. As a side effect, the behavior of those processors can be difficult to predict, which makes it hard to guarantee a certain latency limit on neural network inference. In contrast, TPU design is strictly minimal and deterministic as it has to run only one task at a time: neural network prediction. You can see its simplicity in the floor plan of the TPU die.

61.   Each TPUv2 and TPUv3 Device has "*a computing device adapted to control the operation of the at least one first LPHDR execution unit,*" and that computing device *"comprises at least … a central processing unit (CPU)."* The computing device is the CPU running the Host VM, which is the "master" VM that runs the controlling software program (e.g., as set out below, "your Python code," "your training job", or "machine learning workloads"). The controlling software program "drives the TensorFlow server" (i.e., a TPU VM) which runs on a "TPU worker," or "TPU accelerator," which is a TPUv2 or TPUv3 Device. As explained above, a TPUv2 or TPUv3 Device includes LPHDR execution units. As published by Google:

A TPU training job runs on a two-VM configuration. One VM (the master) runs your Python code. The master drives the TensorFlow server running on a TPU worker.

To use a TPU with AI Platform, configure your training job to access a TPU-enabled machine in one of three ways:

- Use the `BASIC_TPU` scale tier. You can use this method to access TPU v2 accelerators.
- Use a `cloud_tpu` worker and a legacy machine type for the master VM. You can use this method to access TPU v2 accelerators.
- Use a `cloud_tpu` worker and a Compute Engine machine type for the master VM. You can use this method to access TPU v2 or TPU v3 accelerators. TPU v3 accelerators are available in beta.

Basic TPU-enabled machine

Set the scale tier to `BASIC_TPU` to get a master VM and a TPU VM including one TPU with eight TPU v2 cores, as you did when running the previous sample.

Cloud Tensor Processing Units (TPUs)

Tensor Processing Units (TPUs) are Google's custom-developed application-specific integrated circuits (ASICs) used to accelerate machine learning workloads. TPUs are designed from the ground up with the benefit of Google's deep experience and leadership in machine learning.
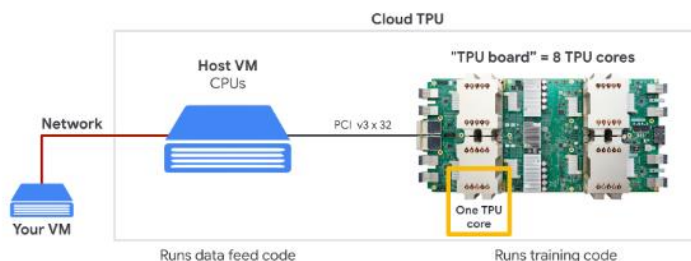
Cloud TPU enables you to run your machine learning workloads on Google's TPU accelerator hardware using TensorFlow. Cloud TPU is designed for maximum performance and flexibility to help researchers, developers, and businesses to build TensorFlow compute clusters that can leverage CPUs, GPUs, and TPUs. High-level Tensorflow APIs help you to get models running on the Cloud TPU hardware.

Advantages of TPUs

Cloud TPU resources accelerate the performance of linear algebra computation, which is used heavily in machine learning applications. TPUs minimize the time-to-accuracy when you train large, complex neural network models. Models that previously took weeks to train on other hardware platforms can converge in hours on TPUs.

Cloud TPU

When you request one "Cloud TPU v2" on Google Cloud Platform, you get a virtual machine (VM) which has a PCI-attached TPU board. The TPU board has four dual-core TPU chips. Each TPU core features a VPU (Vector Processing Unit) and a 128x128 MXU (MatriX multiply Unit). This "Cloud TPU" is then usually connected through the network to the VM that requested it. So the full picture looks like this:

62.   In each TPUv2 and each TPUv3 Device, *"the number of LPHDR execution units in the device exceeds by at least one hundred the non-negative integer number of execution units in the device adapted to execute at least the operation of multiplication on floating point numbers that are at least 32 bits wide."* As shown above, a TPUv2 Device has 8 MXUs, a TPUv3 has 16 MXUs, and each MXU has 16,384 MXU Reduced Precision Multiply Cells. As also shown above, each MXU Reduced Precision Multiply Cell is an LPHDR execution unit that performs a float32 multiplication operation at "reduced bfloat16 precision." Therefore, each TPUv2 Device has at least 131,072 MXU Reduced Precision Multiply Cells that are each LPHDR execution units, and each TPUv3 has at least 262,144 MXU Reduced Precision Multiply

Cells that are each LPHDR execution units. By contrast, there are far fewer execution units adapted to execute the operation of multiplication on floating point numbers that are at least 32 bits wide, on TPUv2 or TPUv3 Devices, since:

- the MXU delivers the "bulk" of a TPU Device's overall arithmetic computation capability. As published by Google:

  Cloud TPU
  ## System Architecture
  Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better training and model accuracy than the IEEE half-precision representation.

- the MXU Reduced Precision Multiply Cells, in performing float32 multiplication operations at "reduced bfloat16 precision," do not "*execute the operation of multiplication on floating point numbers that are at least 32 bits wide.*" Arithmetic (including multiplication) on floating point numbers that are at least 32 bits wide is defined in Singular's patents as "high dynamic range arithmetic of traditional precision." The float32 multiplication operation of an MXU Reduced Precision Multiply Cell is performed at "reduced bfloat16 precision," which entails performing multiplication on floating point numbers that are 16 bits wide. Therefore, the MXU Reduced Precision Multiply Cells themselves do not "*execute the operation of multiplication on floating point numbers that are at least 32 bits wide.*" As published by Google:

  around 4000 cores. A TPU on the other hand uses the bare minimum of hardware for the compute units in the MXU: just `bfloat16 x bfloat16 => float32` multiply-accumulators, nothing else. These are so small that a TPU can implement 16K of them in a 128x128 MXU and process this matrix multiplication in one go.

Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication unit (MXU),   a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then automatically converts them to bfloat16 before calculation. (A TPU can perform FP32 multiplications via multiple iterations of the MXU.) Inside the MXU, multiplications are performed in bfloat16 format, while accumulations are performed in full FP32 precision.
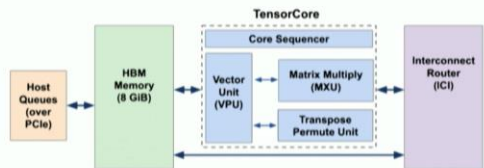
Cloud TPU

System Architecture

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better training and model accuracy than the IEEE half-precision representation.

As affirmed by Dr. Patterson of Google at a lecture presented at the MIT Computer Science and Artificial Intelligence Lab on October 16, 2019, there is roughly a 10:1 ratio of multipliers in MXU versus VPU. The number of MXU Reduced Precision Multiply Cells (i.e., LPHDR execution units, as shown above) in each TPUv2 Device and TPUv3 Device (i.e., 131,072 for a TPUv2 Device or 262,144 for a TPUv3 Device) far exceeds the number of execution units in those same TPUv2 and TPUv3 Devices adapted to execute multiplication on floating point numbers that are at least 32 bits wide (roughly 13,107 for a TPUv2 Device, and roughly 26,214 for a TPUv3 Device). As published by Google:

MXU and VPU

A TPU v2 core is made of a Matrix Multiply Unit (MXU) which runs matrix multiplications and a Vector Processing Unit (VPU) for all other tasks such as activations, softmax, etc. The VPU handles float32 and int32 computations. The MXU on the other hand operates in a mixed precision 16-32 bit floating point format.

TPUv2 Block Diagram

- A 128x128 systolic **Matrix Multiply Unit (MXU)** performs Nx128x128 matrix multiplications (peak: 32K ops/clock)
- **Transpose Reduction Permute Unit (TRP)** on 128x128 matricies

- **Vector Processing Unit (VPU)** 32 2D Vector registers **Vregs** + 2D Vector memory **Vmem** (16MiB)
  - ≈1/10th performance MXU
- **Core Sequencer** fetches instructions from Instruction Memory **Imem**
- **Inter-Core Interconnect (ICI)** sends messages between TensorCores
- **High Bandwidth Memory (HBM)** interposer with 2 HBM stacks / TC
  - 32 64-bit busses (20x TPUv1)
- Connects to host CPU via PCIe Gen3 x16 bus using **Host DMA Queues**

63.   In knowingly adopting Dr. Bates' patented computer architectures, Google reaps

the very same benefits that were predicted by Dr. Bates in his patent application more than 10

years ago. As published by Google and predicted by Dr. Bates in his patent application:

Choosing bfloat16

Our hardware teams chose bfloat16 for Cloud TPUs to improve hardware efficiency while maintaining the ability to train accurate deep learning models, all with minimal switching costs from FP32. The physical size of a hardware multiplier scales with the *square* of the mantissa width. With fewer mantissa bits than FP16, the bfloat16 multipliers are about half the size in silicon of a typical FP16 multiplier, and they are *eight times* smaller than an FP32 multiplier!

PEs implemented according to certain embodiments of the present invention may be relatively small for PEs that can do arithmetic. This means that there are many PEs per unit of resource (e.g., transistor, area, volume), which in turn means that there is a large amount of arithmetic computational power per unit of resource. This enables larger problems to be solved with a given amount of resource than does traditional computer designs. For instance, a digital embodiment of the present invention built as a large silicon chip fabricated with current state of the art technology might perform tens of thousand of arithmetic operations per cycle, as opposed to hundreds in a conventional GPU or a handful in a conventional multicore CPU. These ratios reflect an architectural advantage of embodiments of the present invention that should persist as fabrication technology continues to improve, even as we reach nanotechnology or other implementations for digital and analog computing.

64.   As a result of Google's infringement of the '156 patent, Singular has suffered

damages in an amount to be determined at trial.

COUNT III
(Google's Infringement of United States Patent No. 10,416,961)

65.   Paragraphs [1-64] are reincorporated by reference as if fully set forth herein.

66.   The '961 patent addresses, *inter alia*, the aforementioned technological problem

of a computer making inefficient use of its transistors.

67.   The '961 patent teaches a technological solution to this problem in the form of

novel, unconventional and counterintuitive computer architectures that include, *inter alia*, the

following:

(i)   At least one LPHDR execution unit (e.g., a processing element) that:

a. accepts input signals representing numerical values, that each have a dynamic

range that is at least as wide as from 1,000,000 to 1/1,000,000, and

b. produces output signals representing numerical values, in response to

requested arithmetic operations, which differ by at least 0.2% from their

respective exact results for at least 10% of all possible valid such requested

operations; and,

(ii)   a computing device adapted to control the operation of the one or more LPHDR

execution units.

Dr. Bates' patented architectures solve the aforementioned problem of inefficient transistor use.

Dr. Bates' LPHDR processing elements utilize a far smaller number of transistors per operation

than the high-precision processing elements of conventional computer architectures. This

difference in the required number of transistors per operation creates the opportunity to pack into

a computer having a normal number of transistors (e.g., several billion transistors) a very large

number of LPHDR processing elements that can collectively perform a number of multiplication

operations per period that is many multiples larger than the number of multiplication operations

per period provided by a conventionally architected computer having similar physical

characteristics (i.e., in terms of its number of transistors, its semiconductor fabrication process,

power draw, etc.).   A computer utilizing Dr. Bates' novel architecture achieves the advantage of

executing a far larger number of operations per period than a conventional computer while

supporting software programs that require operations to be performed on numbers having high

dynamic range.

68.   Dr. Bates' architectural solution to the aforementioned problem of inefficient

transistor usage represented a fundamentally new and unconventional approach to computer

architecture. Dr. Bates' patented architecture was not obvious, or conventional to one of

ordinary skill in the art at the time of the invention. Conventional computer architectures, for

example, even when intended for execution of AI software programs, did not include the concept

of a computer based on low precision high dynamic range (LPHDR) processing elements.

Computer architects even taught away from this concept. Prior to Dr. Bates' invention, such a

computer did not exist.

69.   Computer architects as of 2009 taught away from Dr. Bates' invention. Dr.

Bates' LPHDR processing elements frequently generate, in response to requests to perform

arithmetic operations on high dynamic range numbers, results that materially differ from the

exact results of those operations. For example, the '961 patent teaches an LPHDR processing

element that can generate results in response to requested arithmetic operations that differ by at

least 0.2% from their respective exact results, for at least 10% of all possible valid such

requested operations. It would have been counterintuitive to those in the art as of 2009, to

architect a computer from even one such frequently inexact LPHDR processing element, let

alone a large number of LPHDR processing elements that were all frequently generating such

materially inexact results for a given software program. It was counterintuitive to architect a

computer comprising numerous LPHDR processing elements that each frequently generate

inexact results, knowing that such a computer was going to be used by software programs to

execute many tasks that each required hundreds, thousands or even millions of sequential

arithmetic operations that could accumulate errors. Dr. Bates nonetheless conceived, made and

patented a working computer utilizing such LPHDR processing elements.

70.   The '961 patent ushered in a revolutionary increase in computer efficiency

through improved computer architectures. Its claims specify architectural features pertaining to

a computer design, including LPHDR processing elements, and a computing device that is

adapted to control the operation of the LPHDR execution units.

71.   Google has directly infringed, and continues to directly infringe, literally and/or

by the doctrine of equivalents, at least claims 1-5, 10, 13, 14 and 15 of the '961 patent by

making, using, testing, selling, offering for sale and/or importing into the United States its

TPUv2 and TPUv3 Devices alone or in combination with its existing data servers. Google's

computer systems that infringe the '961 patent include the TPUv2 and TPUv3 Devices. The

infringing TPUv2 and TPUv3 Devices, in Google's own words, "powers" at least Google

Translate, Photos, Search, Assistant, and Gmail. As published by Google:

Empowering businesses with Google Cloud AI

Machine learning has produced business and research breakthroughs ranging from network security to medical diagnoses. We built the Tensor Processing Unit (TPU) in order to make it possible for anyone to achieve similar breakthroughs. Cloud TPU is the custom-designed machine learning ASIC that powers Google products like Translate, Photos, Search, Assistant, and Gmail. Here's how you can put the TPU and machine learning to work accelerating your company's success, especially at scale.

72.   Claim 4 of the '961 patent is reproduced below:

A device comprising:
at least one first low precision high-dynamic range (LPHDR) execution
unit adapted to execute a first operation on a first input signal representing a first
numerical value to produce a first output signal representing a second numerical
value,
wherein the dynamic range of the possible valid inputs to the first operation is at
least as wide as from 1/1,000,000 through 1,000,000 and for at least X=10% of
the possible valid inputs to the first operation, the statistical mean, over repeated
execution of the first operation on each specific input from the at least X % of the
possible valid inputs to the first operation, of the numerical values represented by
the first output signal of the LPHDR unit executing the first operation on that
input differs by at least Y=0.2% from the result of an exact mathematical
calculation of the first operation on the numerical values of that same input;
at least one first computing device adapted to control the operation of the at least
one first LPHDR execution unit;

73.   A TPUv2 or a TPUv3 Device are examples of a "*device*," as claimed by the '961

patent. As published by Google:

TPU versions

Each TPU version defines the specific hardware characteristics of a TPU device. The TPU version
defines the architecture for each TPU core, the amount of high-bandwidth memory (HBM) for each
TPU core, the interconnects between the cores on each TPU device, and the networking interfaces
available for inter-device communication.

Cloud TPU

When you request one "Cloud TPU v2" on Google Cloud Platform, you get a virtual machine (VM) which has a PCI-
attached TPU board. The TPU board has four dual-core TPU chips. Each TPU core features a VPU (Vector Processing
Unit) and a 128x128 MXU (MatriX multiply Unit). This "Cloud TPU" is then usually connected through the network to the
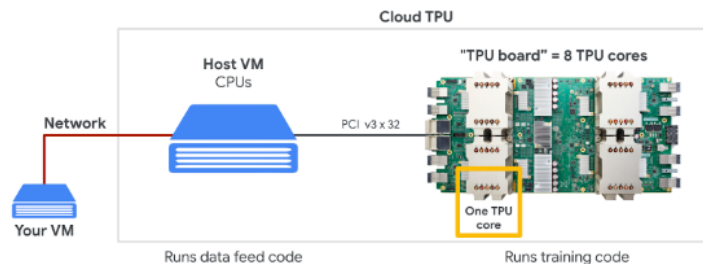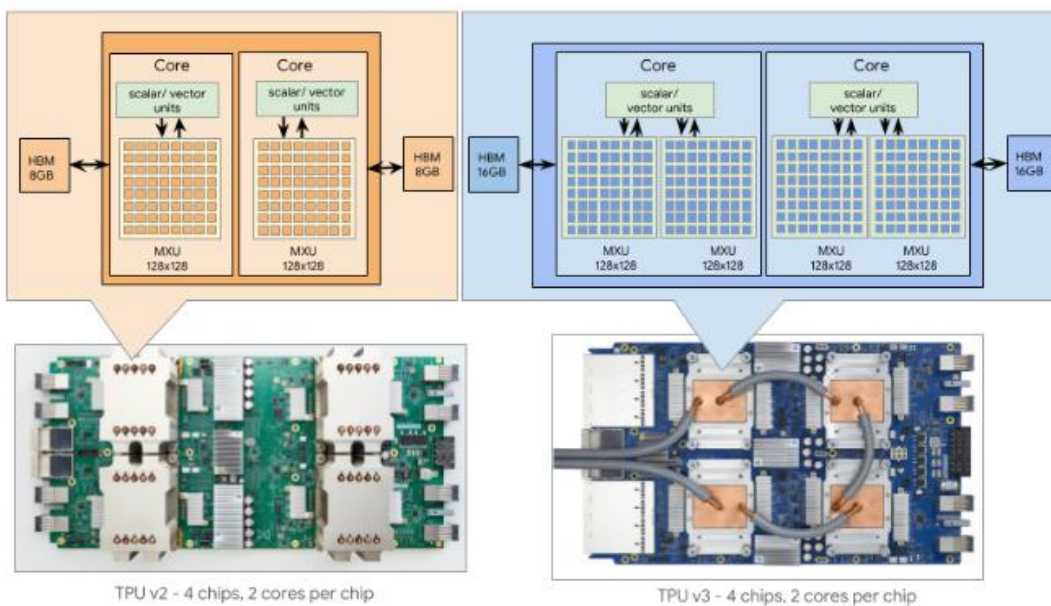VM that requested it. So the full picture looks like this:

Illustration: your VM with a network-attached "Cloud TPU" accelerator. "The Cloud TPU" itself is made of a VM with a PCI-
attached TPU board with four dual-core TPU chips on it.

74.   Each TPUv2 Device and TPUv3 Device infringes claim 4 of the '961 patent, by

inter alia, including multiple matrix multiplication unit (MXU) ALUs and associated circuitry.

a.   Each TPUv2 Device has 8 MXUs (one MXU per TPU core, 2 TPU cores per

chip, and 4 chips per TPUv2 Device), and each TPUv3 Device has 16 MXUs (two

MXUs per TPU core, 2 TPU cores per chip, and 4 chips per TPUv3 Device). As

published by Google:

- TPU v2:
    - 8 GiB of HBM for each TPU core
    - One MXU for each TPU core
    - Up to 512 total TPU cores and 4 TiB of total memory in a TPU Pod
- TPU v3:
    - 16 GiB of HBM for each TPU core
    - Two MXUs for each TPU core
    - Up to 2048 total TPU cores and 32 TiB of total memory in a TPU Pod

TPU v2 - 4 chips, 2 cores per chip   TPU v3 - 4 chips, 2 cores per chip

b.   Each MXU contains a systolic array having 128 X 128 MXU ALUs (i.e., 16,384

ALUs). Therefore, a TPUv2 Device has 131,072 MXU ALUs and a TPUv3

Device has 262,144 MXU ALUs. As published by Google:

> Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication unit (MXU), a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then automatically converts them to bfloat16 before calculation. (A TPU can perform FP32 multiplications via multiple iterations of the MXU.) Inside the MXU, multiplications are performed in bfloat16 format, while accumulations are performed in full FP32 precision.

c.   Each of those one-hundred-thousand-plus ALUs is adapted to execute a multiplication operation on a value that was converted to a "bfloat16" format after being taken as input in 32-bit floating point format ("FP32 format" or "float32"). The circuitry for taking a float32 input signal, converting it to a bfloat16 value, and then multiplying the value, is hereinafter an "MXU Reduced Precision Multiply Cell." An MXU Reduced Precision Multiply Cell comprises the part of an MXU ALU that performs a multiplication operation, and circuitry for taking a float32 input signal and converting it to a bfloat16 value. As published by Google:

> Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication unit (MXU), a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then automatically converts them to bfloat16 before calculation. (A TPU can perform FP32 multiplications via multiple iterations of the MXU.) Inside the MXU, multiplications are performed in bfloat16 format, while accumulations are performed in full FP32 precision.

Cloud TPU
System Architecture

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better training and model accuracy than the IEEE half-precision representation.

75.   Over 100,000 individual MXU Reduced Precision Multiply Cells are in each TPUv2 and TPUv3 Device. Each individual MXU Reduced Precision Multiply Cell is a "*low precision high-dynamic range (LPHDR) execution unit adapted to execute a first operation on a*

first input signal representing a first numerical value to produce a first output signal

representing a second numerical value." The "*first operation*" executed by each individual

MXU Reduced Precision Multiply Cell is a multiplication operation that is (i) performed on two

input signals each representing a float32 numerical value, but (ii) carried out at "reduced

bfloat16 precision." Such an operation (e.g., "X[2,0]*W[0,0]" in the example equation for

Y[2,0] that Google provides below) is a part of a larger float32 matrix multiplication operation

(e.g., Y= X*W in the example Google provides below) being performed at "reduced bfloat16

precision" by the MXU as a whole. As published by Google:

Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication unit (MXU), a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then automatically converts them to bfloat16 before calculation. (A TPU can perform FP32 multiplications via multiple iterations of the MXU.) Inside the MXU, multiplications are performed in bfloat16 format, while accumulations are performed in full FP32 precision.

Systolic array

The MXU implements matrix multiplications in hardware using a so-called "systolic array" architecture in which data elements flow through an array of hardware computation units. (In medicine, "systolic" refers to heart contractions and blood flow, here to the flow of data.)

The basic element of a matrix multiplication is a dot product between a line from one matrix and a column from the other matrix (see illustration at the top of this section). For a matrix multiplication Y=X*W, one element of the result would be:

$$Y[2,0] = X[2,0]*W[0,0] + X[2,1]*W[1,0] + X[2,2]*W[2,0] + ... + X[2,n]*W[n,0]$$
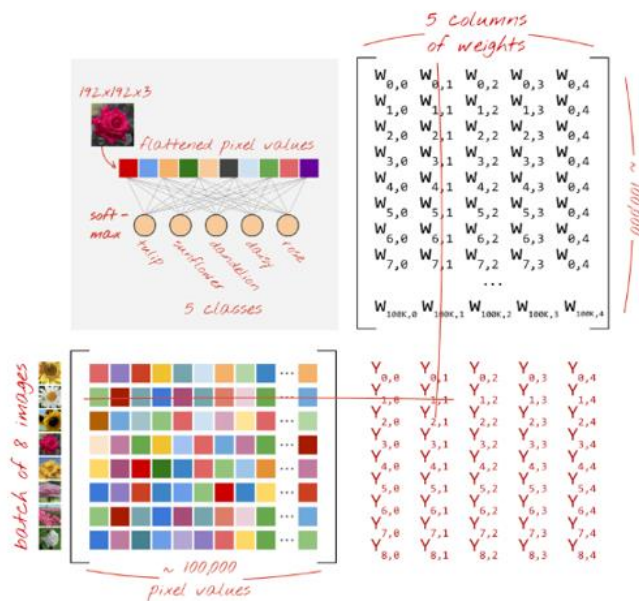
Illustration: a dense neural network layer as a matrix multiplication, with a batch of eight images processed through the neural network at once. Please run through one line x column multiplication to verify that it is indeed doing a weighted sum of all the pixels values of an image. Convolutional layers can be represented as matrix multiplications.

Cloud TPU

System Architecture

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better training and model accuracy than the IEEE half-precision representation.

76.   The "*first input signal*" for each individual MXU Reduced Precision Multiply Cell is the signal representing a float32 value from inside either matrix that is being multiplied by the MXU. In the illustration below taken from a Google animation, such input signals are depicted using the red dots moving rightwards into the left of the MXU and its Reduced Precision Multiply Cells. Such input signals are also depicted in the same Google animation using the black/grey dots moving upwards into the bottom of the MXU and its Reduced Precision Multiply Cells. The "*first output signal*" produced by an MXU Reduced Precision Multiply Cell is the result of the float32 multiplication operation it performs at reduced bfloat16 precision. Such a *"first output signal"* flows from the multiplier "x" to the adjacent adder "+" in the illustration below, and is described as a "result" in Google's own words: "As each

multiplication is executed, the result will be passed to the next multipliers while taking

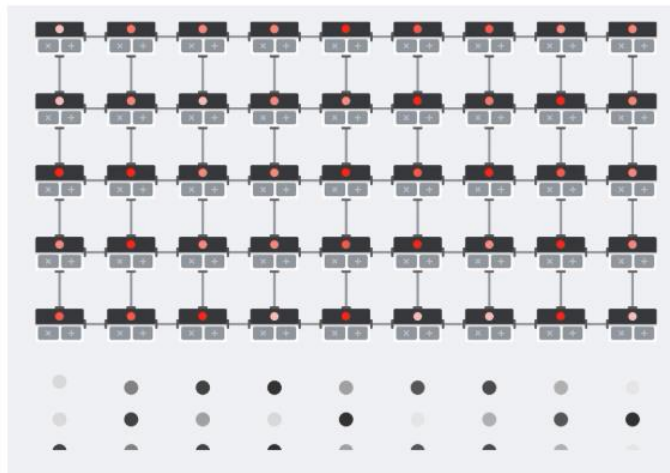summation at the same time." As published by Google:

Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication unit (MXU), a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then automatically converts them to bfloat16 before calculation. (A TPU can perform FP32 multiplications via multiple iterations of the MXU.) Inside the MXU, multiplications are performed in bfloat16 format, while accumulations are performed in full FP32 precision.
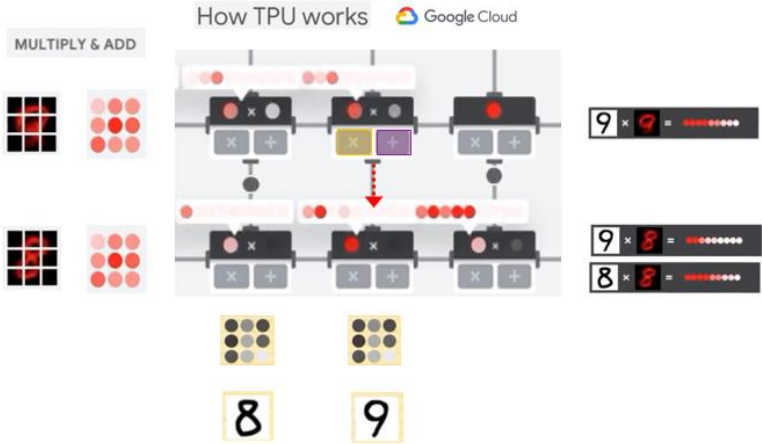
Cloud TPU

System Architecture

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better training and model accuracy than the IEEE half-precision representation.

On a GPU, one would program this dot product into a GPU "core" and then execute it on as many "cores" as are available in parallel to try and compute every value of the resulting matrix at once. If the resulting matrix is 128x128 large, that would require 128x128=16K "cores" to be available which is typically not possible. The largest GPUs have around 4000 cores. A TPU on the other hand uses the bare minimum of hardware for the compute units in the MXU: just `bfloat16 x bfloat16 => float32` multiply-accumulators, nothing else. These are so small that a TPU can implement 16K of them in a 128x128 MXU and process this matrix multiplication in one go.

Illustration: the MXU systolic array. The compute elements are multiply-accumulators. The values of one matrix are loaded into the array (red dots). Values of the other matrix flow through the array (grey dots). Vertical lines propagate the values up. Horizontal lines propagate partial sums. It is left as an exercise to the user to verify that as the data flows through the array, you get the result of the matrix multiplication coming out of the right side.
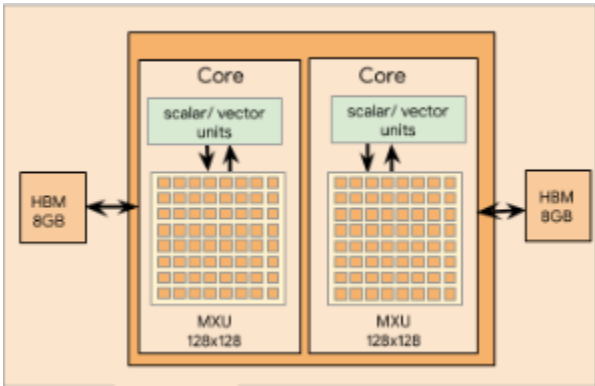
77.   The "*first numerical value*" represented by the "*first input signal*," and the

"*second numerical value*" represented by the "*first output signal*," are all float32 numbers.

Float32 numbers are "*numerical values*." As published by Google:

Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication
unit (MXU),   a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and
multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the
majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then
automatically converts them to bfloat16 before calculation. (A TPU can perform
FP32 multiplications via multiple iterations of the MXU.) Inside the MXU,
multiplications are performed in bfloat16 format, while accumulations are
performed in full FP32 precision.

Cloud TPU
System Architecture

Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute
power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each
cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies
at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better
training and model accuracy than the IEEE half-precision representation.

On a GPU, one would program this dot product into a GPU "core" and then execute it on as many "cores" as are available in parallel to try and compute every value of the resulting matrix at once. If the resulting matrix is 128x128 large, that would require 128x128=16K "cores" to be available which is typically not possible. The largest GPUs have around 4000 cores. A TPU on the other hand uses the bare minimum of hardware for the compute units in the MXU: just `bfloat16 x bfloat16 => float32` multiply-accumulators, nothing else. These are so small that a TPU can implement 16K of them in a 128x128 MXU and process this matrix multiplication in one go.
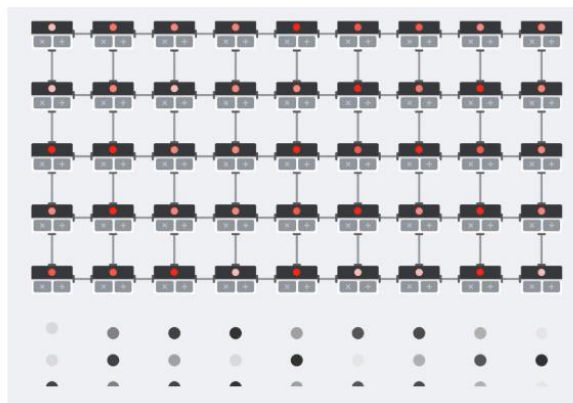

Illustration: the MXU systolic array. The compute elements are multiply-accumulators. The values of one matrix are loaded into the array (red dots). Values of the other matrix flow through the array (grey dots). Vertical lines propagate the values up. Horizontal lines propagate partial sums. It is left as an exercise to the user to verify that as the data flows through the array, you get the result of the matrix multiplication coming out of the right side.

Single-precision floating-point format

From Wikipedia, the free encyclopedia

Single-precision floating-point format is a computer number format, usually occupying 32 bits in computer memory; it represents a wide dynamic range of numeric values by using a floating radix point.

A floating-point variable can represent a wider range of numbers than a fixed-point variable of the same bit width at

78.   Each MXU Reduced Precision Multiply Cell is an "*LPHDR execution unit.*"

Specifically:

- For each MXU Reduced Precision Multiply Cell, "*the dynamic range of the possible valid inputs to the first operation is at least as wide as from 1/1,000,000 through 1,000,000.*" As shown above, each MXU Reduced Precision Multiply Cell performs a float32 multiplication operation at "reduced bfloat16 precision" on valid input signals representing numerical values having a float32 format. A float32 numerical value, whose format is shown below, has the following dynamic range:

  Minimum: $2^{-126} \approx 1.175494351 \times 10^{-38}$

  Maximum: $(2 - 2^{-23}) \times 2^{127} \approx 3.402823466 \times 10^{38}$

  As published by Google:

(a) fp32: Single-precision IEEE Floating Point Format   Range: ~$1e^{-38}$ to ~$3e^{38}$

Exponent: 8 bits   Mantissa (Significand): 23 bits

(b) fp16: Half-precision IEEE Floating Point Format   Range: ~$5.96e^{-8}$ to 65504

Exponent: 5 bits   Mantissa (Significand): 10 bits

(c) bfloat16: Brain Floating Point Format   Range: ~$1e^{-38}$ to ~$3e^{38}$

Exponent: 8 bits   Mantissa (Significand): 7 bits

As Figure 1 shows, bfloat16 has a greater dynamic range—i.e., number of exponent bits—than FP16. In fact, the dynamic range of bfloat16 is identical to that of FP32. We've trained a wide range of deep learning models, and in our experience, the bfloat16 format works as well as the FP32 format while delivering increased performance and reducing memory usage.

- For each MXU Reduced Precision Multiply Cell, "*for at least X=10% of the possible valid inputs to the first operation… the numerical values represented by the first output signal of the LPHDR unit executing the first operation on that input differs by at least Y=0.2% from the result of an exact mathematical calculation of the first operation on the numerical values of that same input.*" Specifically, each TPUv2 and TPUv3 MXU Reduced Precision Multiply Cell performs a float32 multiplication operation but does so in Google's own words at "reduced bfloat16 precision." Each MXU Reduced Precision Multiply Cell takes the following steps: (i) receives as input two signals that each represent a float32 numerical value, (ii) converts each of the received float32 numerical values to a bfloat16 numerical value, (iii) multiplies the resulting pair of bfloat16 numerical values with each other, and (iv) adjusts the format of the result of the bfloat16 multiplication generated in step (iii), if needed, to produce an output signal that represents a float32 numerical value to be accumulated. As published by Google:

58

Cloud TPU v2 and Cloud TPU v3 primarily use bfloat16 in the matrix multiplication unit (MXU),   a 128 x 128 systolic array. There are two MXUs per TPUv3 chip and multiple TPU chips per Cloud TPU system. Collectively, these MXUs deliver the majority of the total system FLOPS. Each MXU takes inputs in FP32 format but then automatically converts them to bfloat16 before calculation. (A TPU can perform FP32 multiplications via multiple iterations of the MXU.) Inside the MXU, multiplications are performed in bfloat16 format, while accumulations are performed in full FP32 precision.

Cloud TPU
System Architecture
Each TPU core has scalar, vector, and matrix units (MXU). The MXU provides the bulk of the compute power in a TPU chip. Each MXU is capable of performing 16K multiply-accumulate operations in each cycle. While the MXU inputs and outputs are 32-bit floating point values, the MXU performs multiplies at reduced bfloat16 precision. Bfloat16 is a 16-bit floating point representation that provides better training and model accuracy than the IEEE half-precision representation.

When the float32 numerical values produced by the TPU's float32 multiplication operation (which, as shown above, is performed at "reduced bfloat16 precision"), for a mathematically representative sample of all possible valid pairs of inputted float32 numerical values, are compared to the numerical values produced by the exact full precision multiplication operations for those same respective valid pairs of inputted float32 numerical values, the TPU's float32 numerical values differ, for at least 10% of those multiplied pairs, from the respective exact full precision values, by at least 0.2%. This is illustrated by the Singular test results shown below.

	bf16
% of valid > 1.00%	4.65%
% of valid > 0.50%	55.39%
% of valid > 0.20%	92.69%
% of valid > 0.10%	98.15%
% of valid > 0.05%	99.52%

- For each MXU Reduced Precision Multiply Cell, *"the statistical mean, over repeated execution of the first operation on each specific input from the at least X % of the possible valid inputs to the first operation, of the numerical values represented by the*

first output signal of the LPHDR unit executing the first operation on that input," will simply equal the numerical value represented by the output signal produced when the MXU Reduced Precision Multiply Cell (i.e., an LPHDR unit) executes an operation on input signals. Each MXU Reduced Precision Multiply Cell is part of a TPUv2 Device or a TPUv3 Device, which are deterministic in their designs (i.e., an operation repeatedly performed by a TPUv2 or a TPUv3 Device on a given set of inputs signals will always yield the same output signal). As published by Google:

> Because general-purpose processors such as CPUs and GPUs must provide good performance across a wide range of applications, they have evolved myriad sophisticated, performance-oriented mechanisms. As a side effect, the behavior of those processors can be difficult to predict, which makes it hard to guarantee a certain latency limit on neural network inference. In contrast, TPU design is strictly minimal and deterministic as it has to run only one task at a time: neural network prediction. You can see its simplicity in the floor plan of the TPU die.

79.   Each TPUv2 and TPUv3 Device has "*a computing device adapted to control the operation of the at least one first LPHDR execution unit.*" The computing device is the CPU running the Host VM, which is the "master" VM that runs the controlling software program (e.g., as set out below, "your Python code," "your training job", or "machine learning workloads"). The controlling software program "drives the TensorFlow server" (i.e., a TPU VM) which runs on a "TPU worker," or "TPU accelerator," which is a TPUv2 or TPUv3 Device. As explained above, a TPUv2 or TPUv3 Device includes LPHDR execution units. As published by Google:

A TPU training job runs on a two-VM configuration. One VM (the master) runs your Python code. The master drives the TensorFlow server running on a TPU worker.

To use a TPU with AI Platform, configure your training job to access a TPU-enabled machine in one of three ways:

- Use the `BASIC_TPU` scale tier. You can use this method to access TPU v2 accelerators.
- Use a `cloud_tpu` worker and a legacy machine type for the master VM. You can use this method to access TPU v2 accelerators.
- Use a `cloud_tpu` worker and a Compute Engine machine type for the master VM. You can use this method to access TPU v2 or TPU v3 accelerators. TPU v3 accelerators are available in beta.

Basic TPU-enabled machine

Set the scale tier to `BASIC_TPU` to get a master VM and a TPU VM including one TPU with eight TPU v2 cores, as you did when running the previous sample.

Cloud Tensor Processing Units (TPUs)

Tensor Processing Units (TPUs) are Google's custom-developed application-specific integrated circuits (ASICs) used to accelerate machine learning workloads. TPUs are designed from the ground up with the benefit of Google's deep experience and leadership in machine learning.
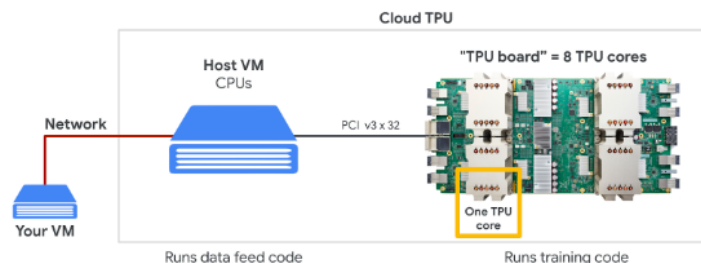
Cloud TPU enables you to run your machine learning workloads on Google's TPU accelerator hardware using TensorFlow ☑. Cloud TPU is designed for maximum performance and flexibility to help researchers, developers, and businesses to build TensorFlow compute clusters that can leverage CPUs, GPUs, and TPUs. High-level Tensorflow APIs help you to get models running on the Cloud TPU hardware.

Advantages of TPUs

Cloud TPU resources accelerate the performance of linear algebra computation, which is used heavily in machine learning applications. TPUs minimize the time-to-accuracy when you train large, complex neural network models. Models that previously took weeks to train on other hardware platforms can converge in hours on TPUs.

Cloud TPU

When you request one "Cloud TPU v2" on Google Cloud Platform, you get a virtual machine (VM) which has a PCI-attached TPU board. The TPU board has four dual-core TPU chips. Each TPU core features a VPU (Vector Processing Unit) and a 128x128 MXU (MatriX multiply Unit). This "Cloud TPU" is then usually connected through the network to the VM that requested it. So the full picture looks like this:

Illustration: your VM with a network-attached "Cloud TPU" accelerator. "The Cloud TPU" itself is made of a VM with a PCI-attached TPU board with four dual-core TPU chips on it.

80.   In knowingly adopting Dr. Bates' patented computer architectures, Google reaps the very same benefits that were predicted by Dr. Bates in his patent application more than 10 years ago. As published by Google and predicted by Dr. Bates in his patent application:

Choosing bfloat16

Our hardware teams chose bfloat16 for Cloud TPUs to improve hardware efficiency while maintaining the ability to train accurate deep learning models, all with minimal switching costs from FP32. The physical size of a hardware multiplier scales with the *square* of the mantissa width. With fewer mantissa bits than FP16, the bfloat16 multipliers are about half the size in silicon of a typical FP16 multiplier, and they are *eight times* smaller than an FP32 multiplier!

> PEs implemented according to certain embodiments of the present invention may be relatively small for PEs that can do arithmetic. This means that there are many PEs per unit of resource (e.g., transistor, area, volume), which in turn means that there is a large amount of arithmetic computational power per unit of resource. This enables larger problems to be solved with a given amount of resource than does traditional computer designs. For instance, a digital embodiment of the present invention built as a large silicon chip fabricated with current state of the art technology might perform tens of thousand of arithmetic operations per cycle, as opposed to hundreds in a conventional GPU or a handful in a conventional multicore CPU. These ratios reflect an architectural advantage of embodiments of the present invention that should persist as fabrication technology continues to improve, even as we reach nanotechnology or other implementations for digital and analog computing.

81.   As a result of Google's infringement of the '961 patent, Singular has suffered damages in an amount to be determined at trial.

PRAYER FOR RELIEF

WHEREFORE, Plaintiff prays that the Court:

A.   enter judgment in favor of the Plaintiff on all counts of the Complaint;

B   award Plaintiff damages as determined at trial;

C.   award Plaintiff treble damages, costs and attorney's fees as a result of Defendant's willful infringement;

D.   enjoin Defendant's infringement; and

E.   award Plaintiffs such other and further legal and equitable relief as the Court may

deem just and proper.

DEMAND FOR JURY TRIAL

Plaintiff demands a trial by jury on all counts of the complaint.

Dated: December 20, 2019   Respectfully submitted,

                                            /s/ Paul J. Hayes
                                            Paul J. Hayes (BBO #227000)
                                            Matthew D. Vella (BBO #660171)
                                            Kevin Gannon (BBO #640931)
                                            Daniel McGonagle (BBO #690084)
                                            Alex Breger (BBO #685537)
                                            PRINCE LOBEL TYE LLP
                                            One International Place, Suite 3700
                                            Boston, MA 02110
                                            Tel: (617) 456-8000
                                            Email: phayes@princelobel.com
                                            Email: mvella@princelobel.com
                                            Email: kgannon@princelobel.com
                                            Email: dmcgonagle@princelobel.com
                                            Email: abreger@princelobel.com

                                            ATTORNEYS FOR THE PLAINTIFF