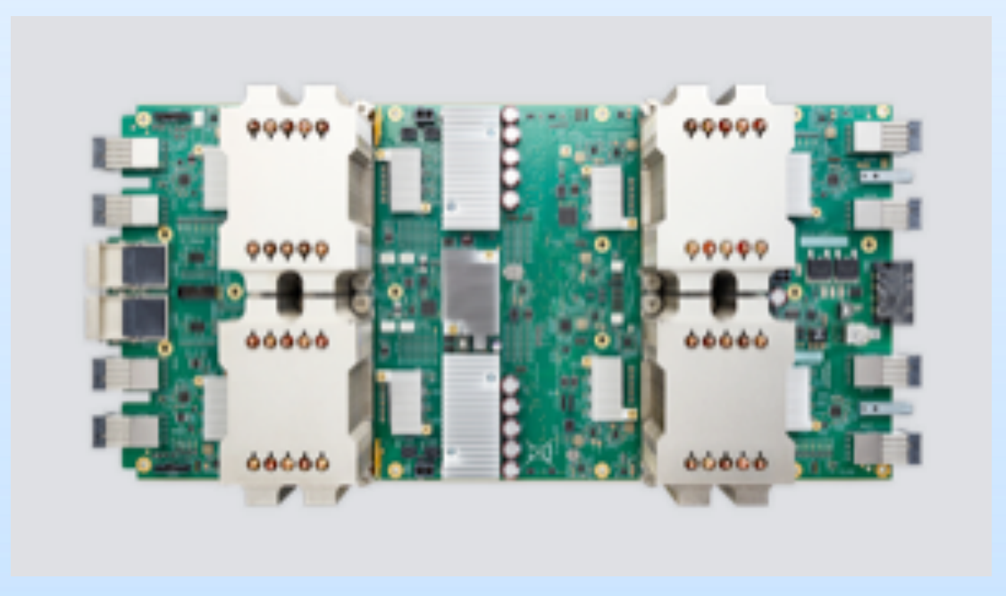
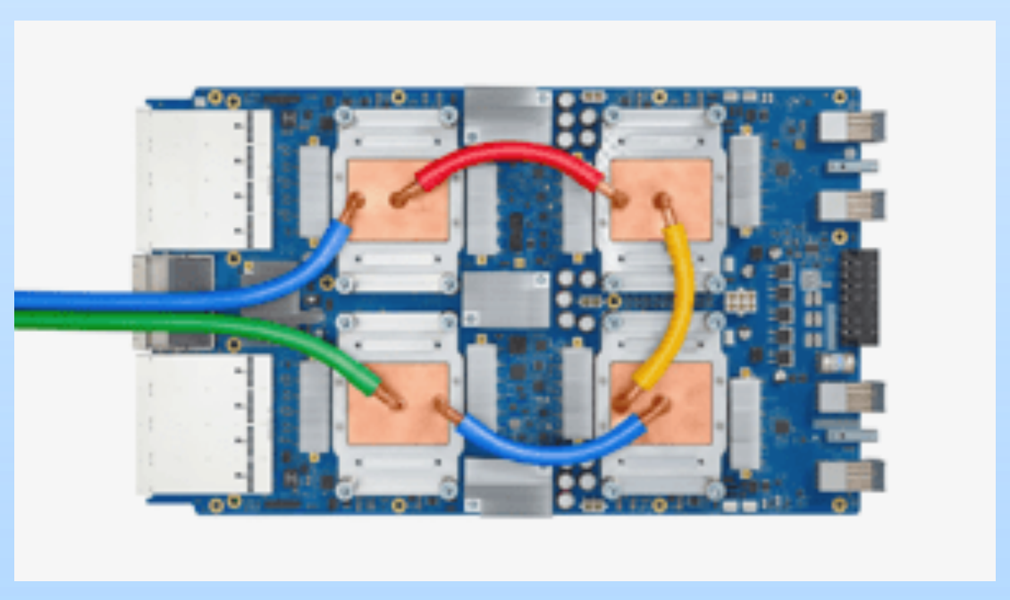
Characterizing Hardware Accelerated Data Center Machine Learning

Abenezer Wudenhe and Hung-Wei Tseng
Extreme Storage & Computer Architecture Laboratory
Department of Electrical and Computer Engineering / Department of Computer Science and Engineering
Bourns College of Engineering, University of California, Riverside

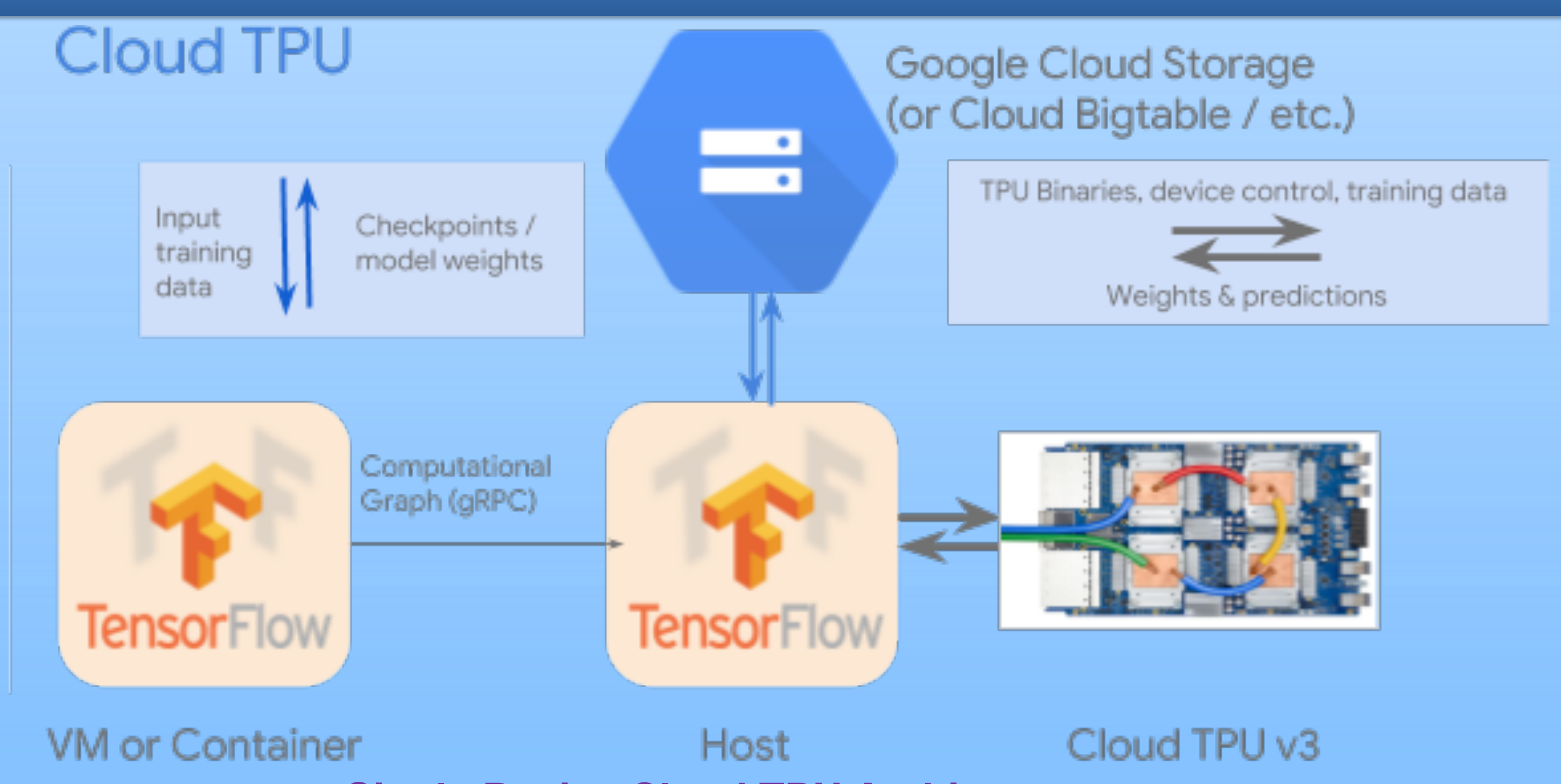
Where are the the new bottlenecks


Cloud TPUv2


Cloud TPUv3

The rise of Machine Learning (ML) applications create growing demand on cloud infrastructures. As popular ML mechanisms heavily rely on matrix operations, conventional general-purpose processors or Graphical Processing Units (GPUs) are optimized for scalar or vector operations, modern computer architectures wastes lots of cycles, power, and energy in performing ML tasks. To accomplish tasks with better energy-efficiency and performance, ML accelerators start gaining ground in datacenters. Google's Tensor Processing Unit (TPUs) offer 70x better performance-per-watt then conventional GPUs. Understanding where new bottle necks lie with accelerated cloud computing becomes critical towards fully utilizing TPUs and other future accelerators.

Cloud TPU Architecture


Cloud TPU
Google Cloud Storage
(or Cloud Bigtable / etc.)
Input training data
Checkpoints / model weights
TPU Binaries, device control, training data
Weights & predictions
Computational Graph (gRPC)
TensorFlow
TensorFlow
VM or Container
Host
Cloud TPU v3
Single Device Cloud TPU Architecture

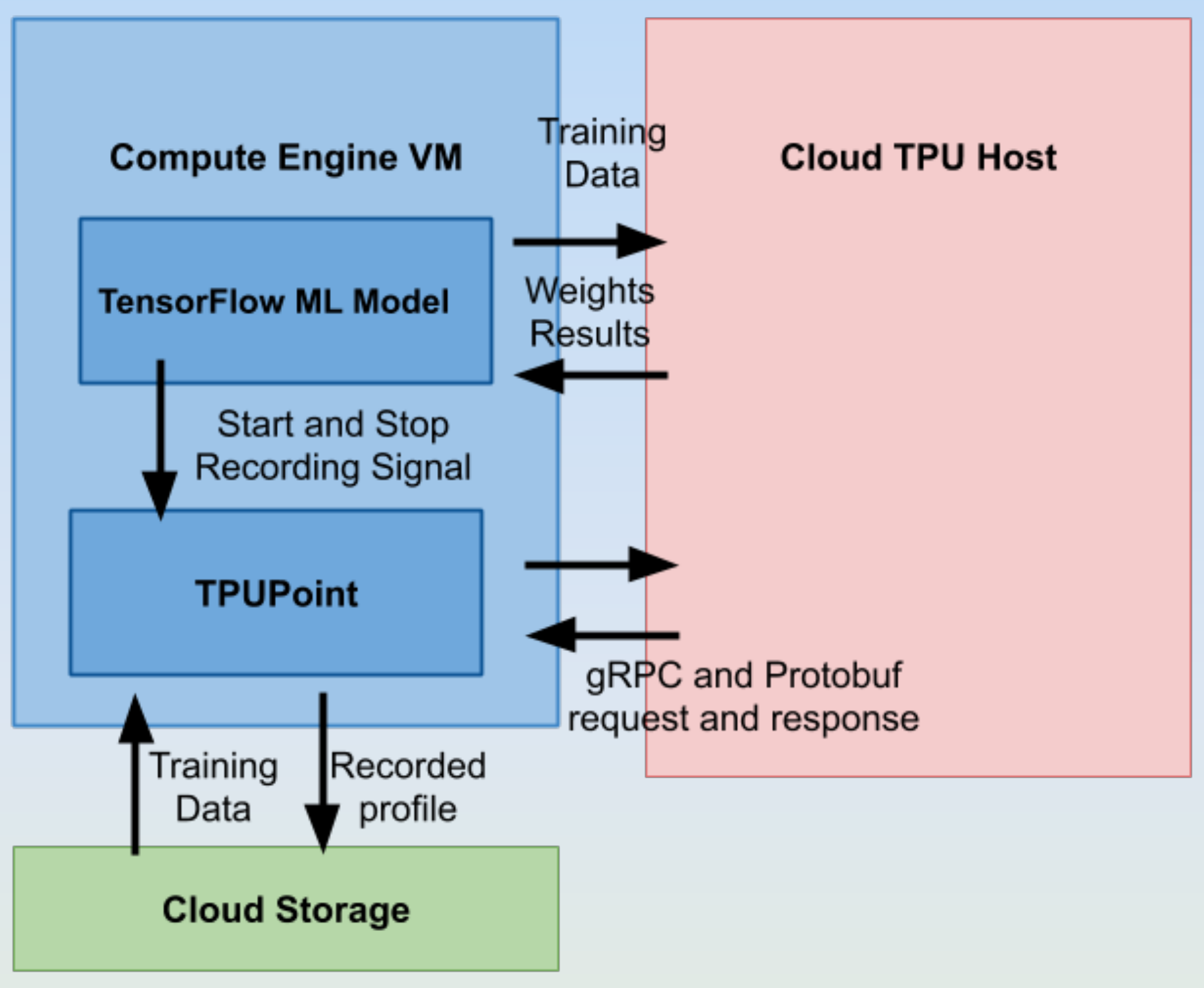
Hardware

Google's cloud platform offers the second and third generation of TPUs (i.e. TPUv2 and TPUv3). TPUv2 contains two cores with one 128x128 Matrix Multiplication Units (MXUs) unit per core and 8 GiB of High Bandwidth Memory (HBM). TPUv3 contains two cores with two 128x128 MXUs per core and 16 GiB of HBM. TPUs are accessible though a cloud Compute Engine Virtual Machine (VM) that range from 2 to 224 CPUs and 2 to 896 GB depending on the chosen configuration. Persistence Storage is provided by Google Cloud though Storage Buckets or Cloud BigTable, where both ML training datasets and model checkpoints are saved. Google organizes 4 TPUs to a single board for 8 cores in total.

Software

Currently, the only established framework available to utilize TPUs is though TensorFlow. TensorFlow makes heavy use of Protocol Buffers (Protobuf) and Google's Remote Procedural Call (gRPC). Protocol Buffers are Google's mechanism for serializing structured data, making it an ideal candidate for both cloud and distributed ML training.

Profiling



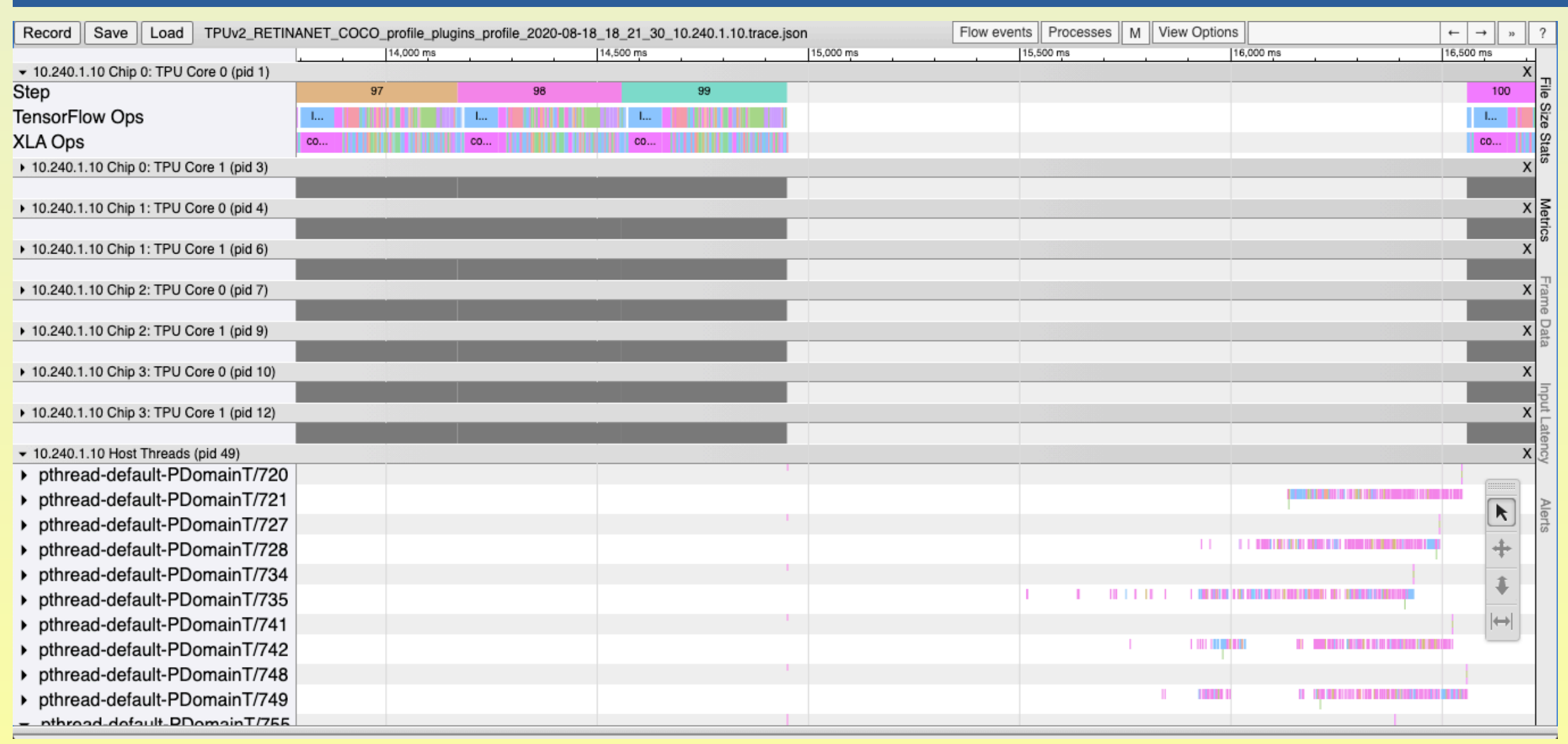
TPUPoint System Architecture

Proposing TPUPoint-Profiler

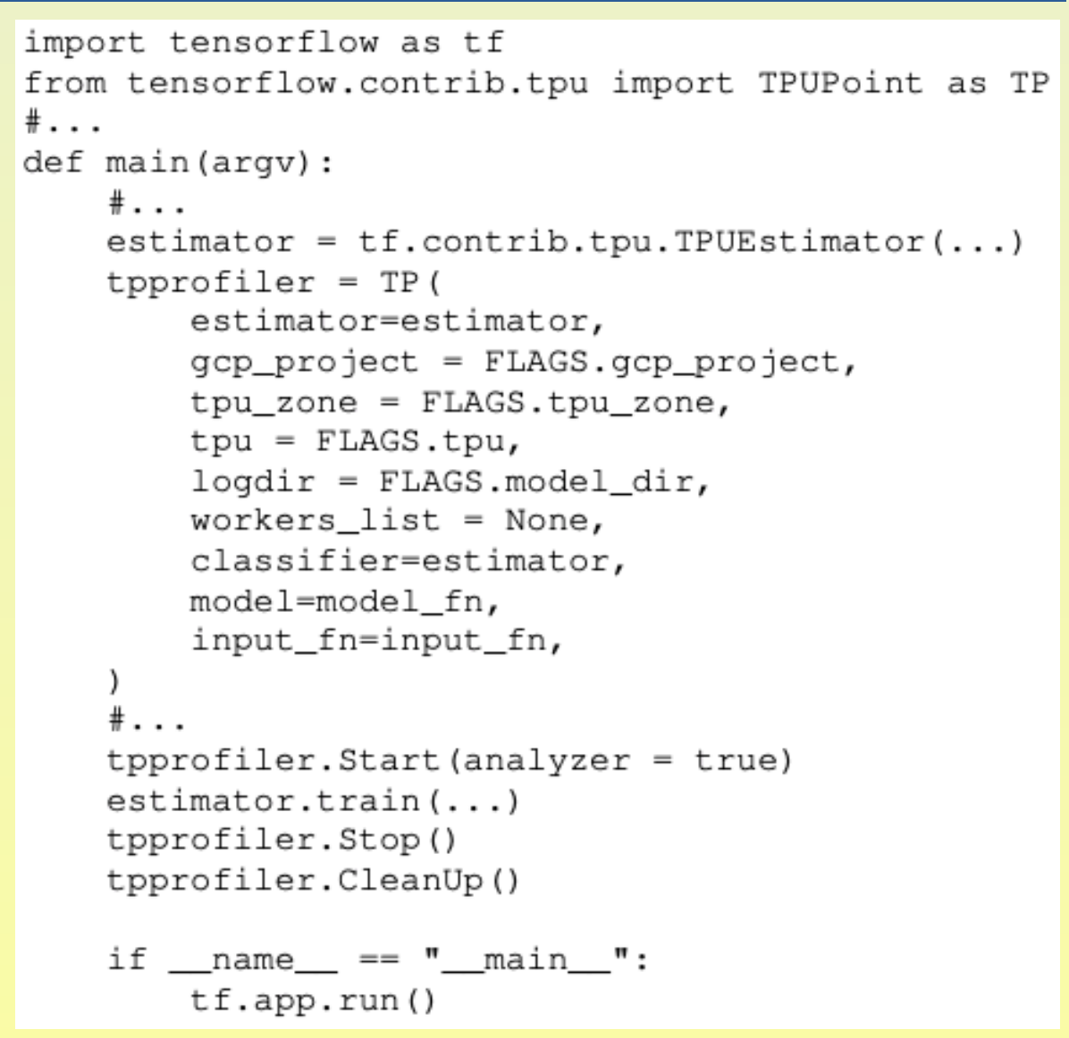
This work proposes TPUPoint to facilitate the process of developing efficient applications of TPU-based applications and identification of new bottleneck operations. Cloud TPU's implementation is not fully available to the public. However registered API calls and serviceable requests have been made available though the `cloud-tpu-profiler` command line tool. This tool utilizes gRPC call to request a profile record of a TPU for small iterations. This tool is limiting in itself as it currently does not profile the entirety of execution of ML applications. TPUPoint provides a programing interface within TensorFlow for users to integrate with ML applications and draw insight as to what operations cause new bottle necks with TPU accelerators.

Profile Record

TPUPoint creates profile requests to the TPU device and receives profile records. A single profile record contains high level TensorFlow operations. The amount of operations recorded depends on the execution of the ML application and the duration of the recording. The default is 60 seconds but can be adjusted by the user. Both TPU and Host operations are stored in a JSON format and can be viewed using `chrome://tracing`, built into the Google Chrome browser.



Single Profile Record

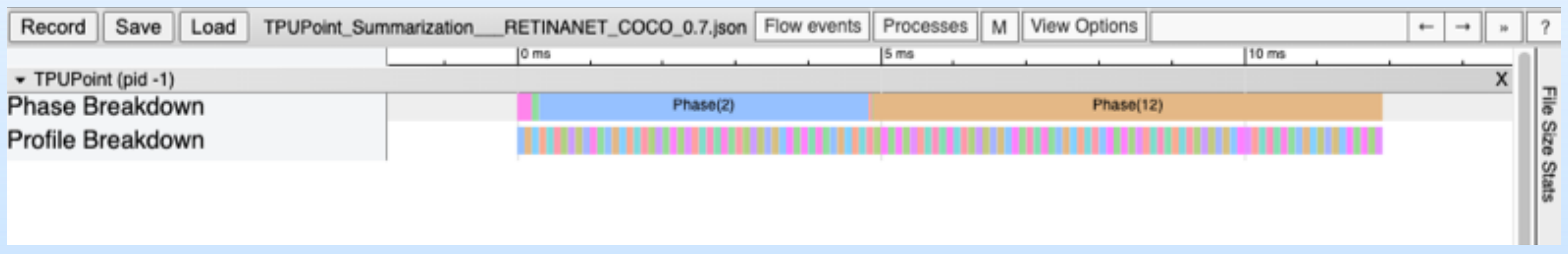


```
import tensorflow as tf
from tensorflow.contrib.tpu import TPUPoint as TP
#...
def main(argv):
    #...
    estimator = tf.contrib.tpu.TPUEstimator(...)
    tpprofiler = TP(
        estimator=estimator,
        gcp_project = FLAGS.gcp_project,
        tpu_zone = FLAGS.tpu_zone,
        tpu = FLAGS.tpu,
        logdir = FLAGS.model_dir,
        workers_list = None,
        classifier=estimator,
        model=model_fn,
        input_fn=input_fn,
    )
    #...
    tpprofiler.Start(analyzer = true)
    estimator.train(...)
    tpprofiler.Stop()
    tpprofiler.CleanUp()

    if __name__ == "__main__":
        tf.app.run()
```

Python Code example

Design of the Analysis (Phase Summarizations)



Profile Record Summarization

TPUPoint can potentially produce a large amounts of profile records due to varying ML application durations. To avoid high computation overhead due to the computation complexity from the dimensions in each record, TPUPoint walks through and summarizes profile records into program phases. Each program phase represents a larger amount of program exaction. This reduces the overhead when conducting analysis of bottleneck operations. TPUPoint implements three methods to summarization; K-Means, DBScan, and an Online Linear method.

K-Means Algorithm

Records are divided by training step durations. Operations that occur between steps are assigned to the next closest step. Steps are then converted to a frequency vector of operations and PCA is applied to reduce to at a maximum of 100 features. Using a k value, steps are organized into clusters that will be considered as a phase of execution during analysis.

DBScan Algorithm

Records are divided by training step durations. Operations that occur between steps are assigned to the next closest step. Steps are then converted to a frequency vector of operations and PCA is applied to reduce to at a maximum of 100 features. With a minimum value of vectors to form a cluster, DBScan is applied to organize into steps into clusters that will be considered as a phase of execution during analysis.

Online Linear Algorithm

During recording, predecessor steps are compared to successor steps based on overlap of operations to produce a percentage of similarity. Given a minimum similarity threshold value, when similarity between steps falls below the threshold, a phase break is indicated. If no break occurs, the entire execution is summarized into one phase.

Results

Workload	Workload Type	Model	Dataset	Dataset Size
BERT	Natural Language	BERT	SQuAD MRPC MNLI CoLA	422.27 MiB 2.85 MiB 1.3 GiB 1.44 MiB
DCGAN	Image Generation	DCGAN	CIFAR10 MNIST	178.87 MiB 56.21 MiB
QaNET	Q/A Natural Language	QaNET	SQuAD	422.27 MiB
RetinaNet	Object Detection	RetinaNet	COCO	48.49 GiB
ResNet	Image Classificaiton	ResNet-50	ImageNet	143.38 GiB

Top Operations

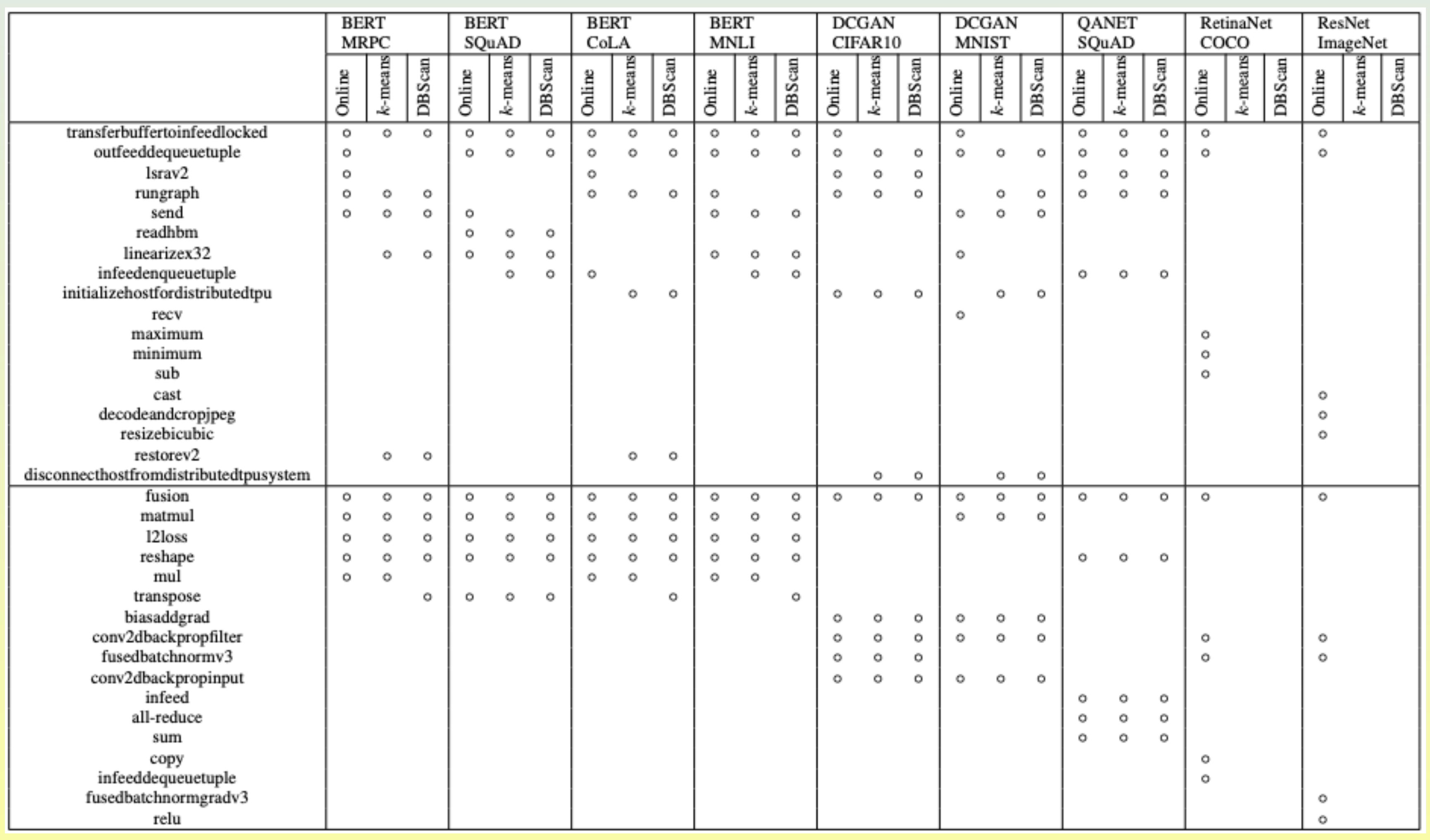
The table above describes the workloads used to identify top operations within a variety of ML applications. The table below shows the top five most time-consuming operations from the top three phases on both the CPU/host and TPU from the application using each of the summarization algorithms. The identified phases are mostly identical and demonstrate the almost same set of top operators across each algorithm We found that all algorithms recognize a common set of most time consuming operators on TPUv2. The top one are; `fusion`, `reshape` and `infeed` type operations.

Computation

Operations such as `fusion` can be considered computational operations. Fusion is created by the XLA compiler and is a combination of compute intensive operations intended to reduce memory transfers

Non-Computation

Operations such as `reshape`, `transpose` and `infeed` can be considered non-computational operations. Often on the host side, these are most critical operations. They are only involved in the transfer or alteration of data into different formats. These operations will continue to grow as TPU and other accelerators improve upon computational operations.



	BERT MRPC			BERT SQuAD			BERT CoLA			BERT MNLI			DCGAN CIFAR10			DCGAN MNIST			QANET SQuAD			RetinaNet COCO			ResNet ImageNet		
	Online	k-means	DBScan	Online	k-means	DBScan	Online	k-means	DBScan	Online	k-means	DBScan	Online	k-means	DBScan	Online	k-means	DBScan	Online	k-means	DBScan	Online	k-means	DBScan	Online	k-means	DBScan
transferbuffertoinfeedlocked	○	○	○	○	○	○	○	○	○	○	○	○	○			○			○	○	○	○			○		
outfeeddequeuetuple	○			○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○			○		
lsrav2	○						○						○	○	○				○	○	○						
rungraph	○	○	○				○	○	○	○			○	○	○		○	○	○	○	○						
send	○	○	○	○						○	○	○				○	○	○									
readhbm				○	○	○																					
linearizex32		○	○	○	○	○				○	○	○				○											
infeedenqueuetuple					○	○	○				○	○							○	○	○						
initializehostfordistributedtpu								○	○				○	○	○		○	○									
recv																○											
maximum																						○					
minimum																						○					
sub																						○					
cast																									○		
decodeandcropjpeg																									○		
resizebicubic																									○		
restorev2		○	○					○	○																		
disconnecthostfromdistributedtpusystem														○	○		○	○									
fusion	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○	○			○		
matmul	○	○	○	○	○	○	○	○	○	○	○	○				○	○	○									
l2loss	○	○	○	○	○	○	○	○	○	○	○	○															
reshape	○	○	○	○	○	○	○	○	○	○	○	○							○	○	○						
mul	○	○					○	○		○	○																
transpose			○	○	○	○			○			○															
biasaddgrad													○	○	○	○	○	○									
conv2dbackpropfilter													○	○	○	○	○	○				○			○		
fusedbatchnormv3													○	○	○							○			○		
conv2dbackpropinput													○	○	○	○	○	○									
infeed																			○	○	○						
all-reduce																			○	○	○						
sum																			○	○	○						
copy																						○					
infeeddequeuetuple																						○					
fusedbatchnormgradv3																									○		
relu																									○		

Top 5 Most Time-Consuming Operations in the Mist consuming Phase

For more information, please visit us on https://www.escalab.org/