Performance analysis of deep neural networks
making the world safe for Skynet

David Levinthal

Microsoft

Azure Cloud Services Infrastructure

Machine learning and Deep Neural Networks

- Machine learning works by building a network of simple computation nodes executing a “output = F(weight*input + bias)” calculation and using known data to find the optimal weights and biases to identify the patterns in the inputs that correlate to the outputs
- The model is trained on tagged data sets (training)
- The trained model can be used to predict the output for untagged input data (inference)
- https://github.com/David-Levinthal/machine-learning

Deep Neural Networks

- Deep Neural Networks (DNN) can be represented as fabrics of nodes, where the nodes represent numerical operations on (multi dimensional) arrays of data

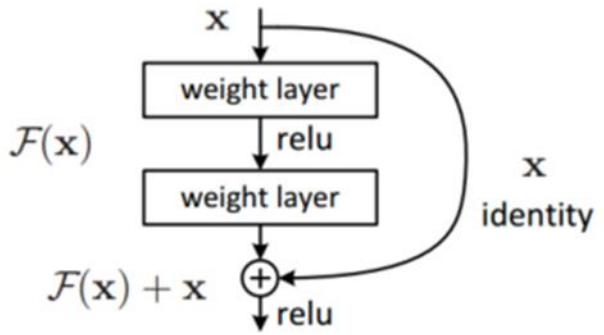
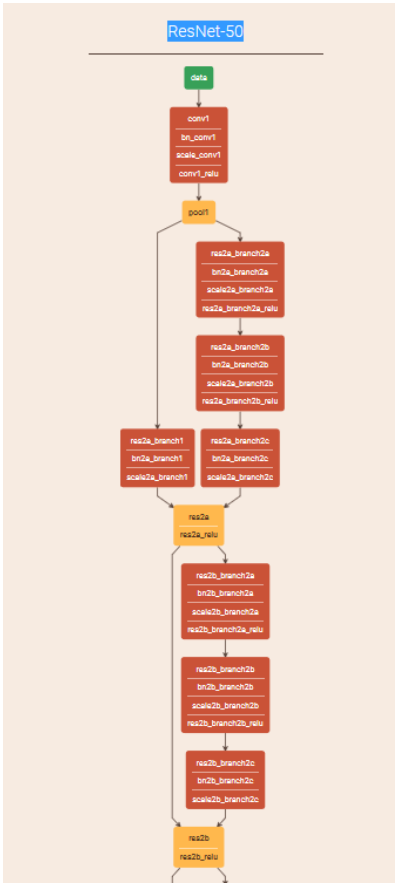
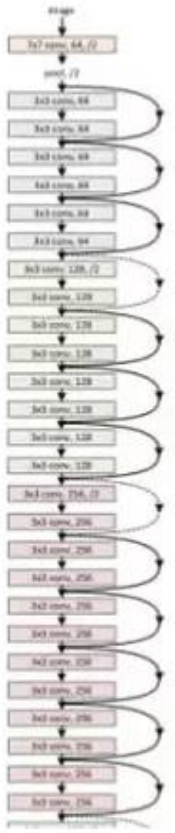


Figure 2. Residual learning: a building block.





Estimating CNN properties III

- Alexnet coded for Tensorflow

```
def __init__(self):
    super(AlexnetModel, self).__init__('alexnet', 224 + 3, 512, 0.005)
 def add_inference(self, cnn):
    # Note: VALID requires padding the images by 3 in width and height
    cnn.conv(64, 11, 11, 4, 4, 'VALID')
    cnn.mpool(3, 3, 2, 2)
    cnn.conv(192, 5, 5)
    cnn.mpool(3, 3, 2, 2)
    cnn.conv(384, 3, 3)
    cnn.conv(384, 3, 3)
    cnn.conv(256, 3, 3)
    cnn.mpool(3, 3, 2, 2)
    cnn.reshape([-1, 256 * 6 * 6])
    cnn.affine(4096)
    cnn.dropout()
    cnn.affine(4096)
    cnn.dropout()
```

Alexnet layer (Soumith/convnet)	expected	measured	ratio
conv (images, 3, 64, 11, 11, 4, 4, 'VALID')	140553600	143717440	1.02251
+ mpool(conv1, 3, 3, 2, 2)	small	46656	
+ conv (pool1, 64, 192, 5, 5, 1, 1, 'SAME')	447897600	472294080	1.054469
+ mpool(conv2, 3, 3, 2, 2)	small	32448	
+ conv (pool2, 192, 384, 3, 3, 1, 1, 'SAME')	224280576	162168191	0.723059
+ conv (conv3, 384, 256, 3, 3, 1, 1, 'SAME')	299040768	215558401	0.720833
+ conv (conv4, 256, 256, 3, 3, 1, 1, 'SAME')	199360512	143927547	0.721946
+ mpool(conv5, 3, 3, 2, 2)	small	9216	
+ tf.reshape(pool5, [-1, 256 * 6 * 6])	small	0	
+ affine(resh1, 256 * 6 * 6, 4096)	75497472	76988416	1.019748
+ affine(affn1, 4096, 4096)	33554432	34226176	1.02002
+ affine(affn2, 4096, 1000)	8192000	8522048	1.040289

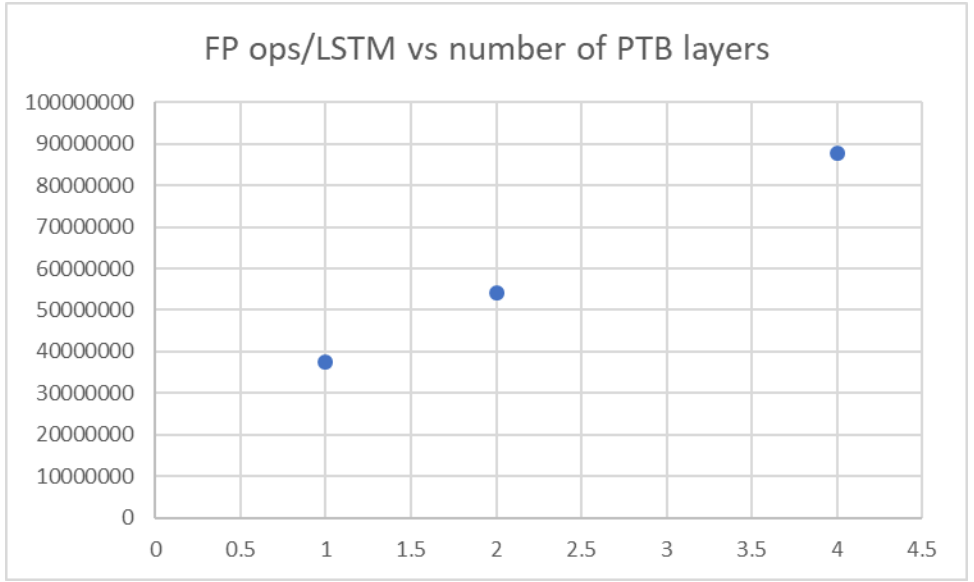
- The 3X3 convolutions here were done with Winograd optimized functions (less than 18 FP ops)
- Measurements were done with NVProf which uses binary instrumentation (165X slow down)

Estimating RNN properties

- LSTM cell is expected to execute 16* hidden_size^2 FP ops
- Penn TreeBank (PTB) test is a simple benchmark predicting next word
- It can have a variable number of layers, hidden size and time steps
  - Set hidden size=1024, time steps=32, batch size=128 and vary layer count

	1 Layer	2 Layers	4 Layers
total fp_ops	2.64E+12	3.82E+12	6.16E+12
Sgemm fp_ops	2.61E+12	3.78E+12	6.12E+12

	sgemm fp ops	sgemm fp opps/LSTM	2 point slope	ratio slope/expected value
1 layer	2.61E+12	3.75E+07		
2 layers	3.78E+12	5.43E+07	16781312	1.000244
4 layers	6.12E+12	8.78E+07	16781312	1.000244



- There is a large non zero baseline

Estimating Transformer Properties

- Built of modules consisting of a multi head attention (8 or 16 heads)
- and a residual feed forward
- With Nx = 6: Total FP ops ~ 6 * (
  3*2* (L^2*dim_model +
  L*dim_model^2/num_head)
  + L* 64 dim_model^2)
- So a Linear term and a quadratic one
- Quadratic term should dominate
  - L ~ 30, dim_model = 1024 or 512

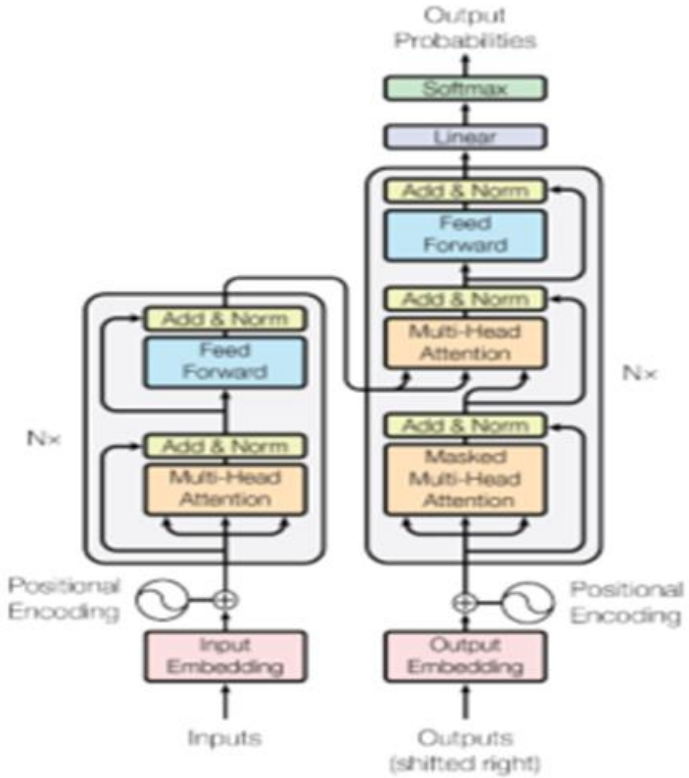
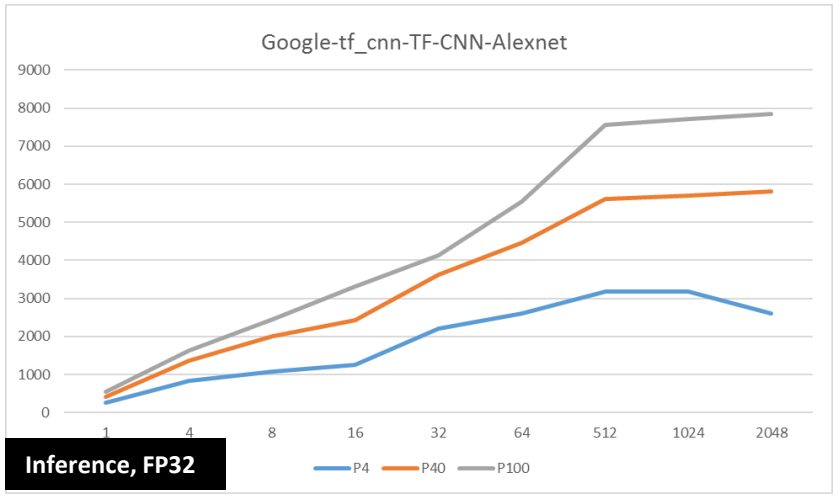


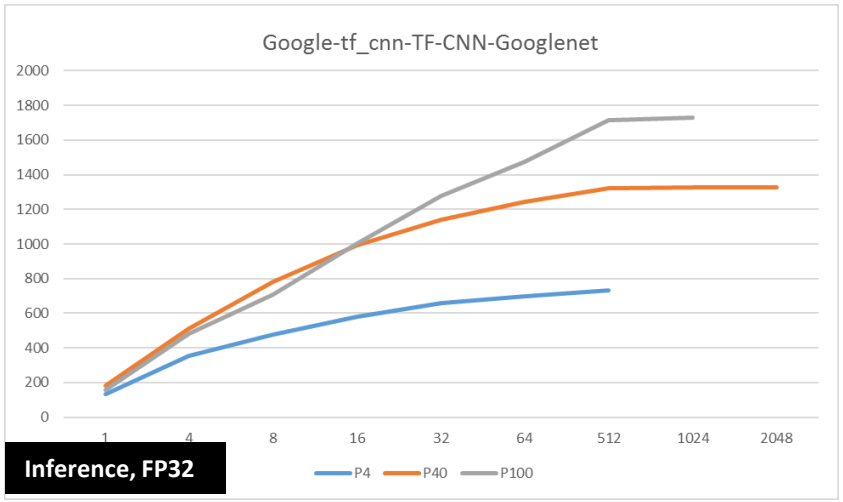
Figure 1: The Transformer - model architecture.

Viewing DNN performance from a hardware perspective

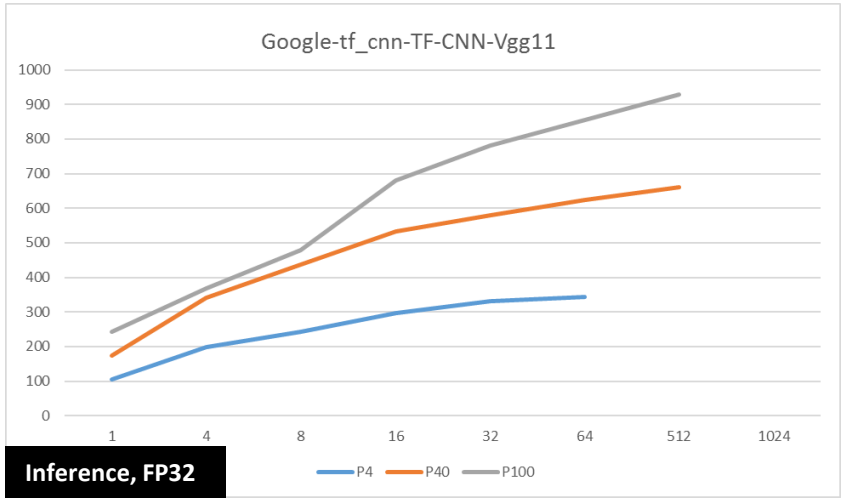
- DNN performance must be separated by training and inference
- In each case there are many run configuration options (hyperparameters)
- Both are effected by minbatch size: number of images/sentences processed together
  - Creates larger matrices for GEMM functions
  - Greatly effects speed
- Training has a many additional options (learning rate, dropout, gradient evaluation, synchronization strategy, etc)

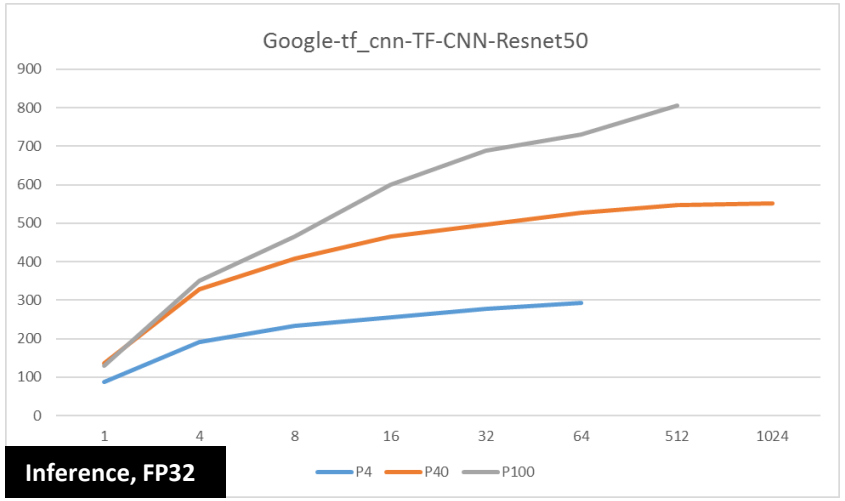
Google TF CNNs – Inference (Images/sec vs. Batch Size)

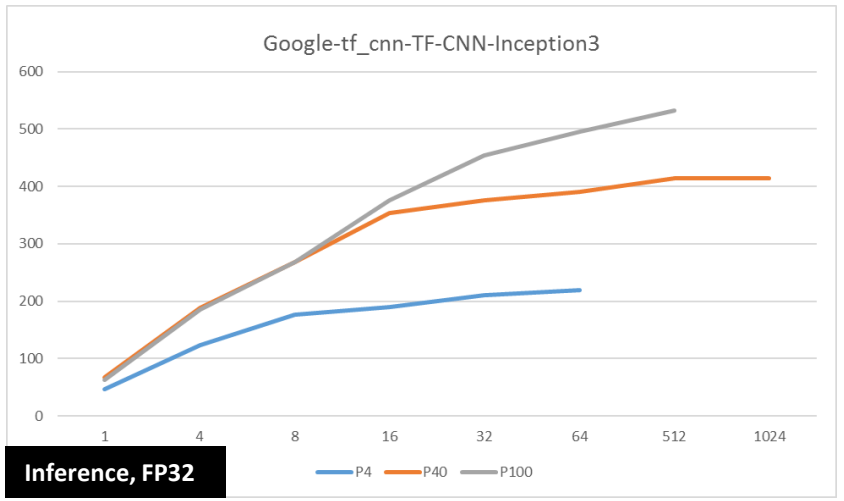




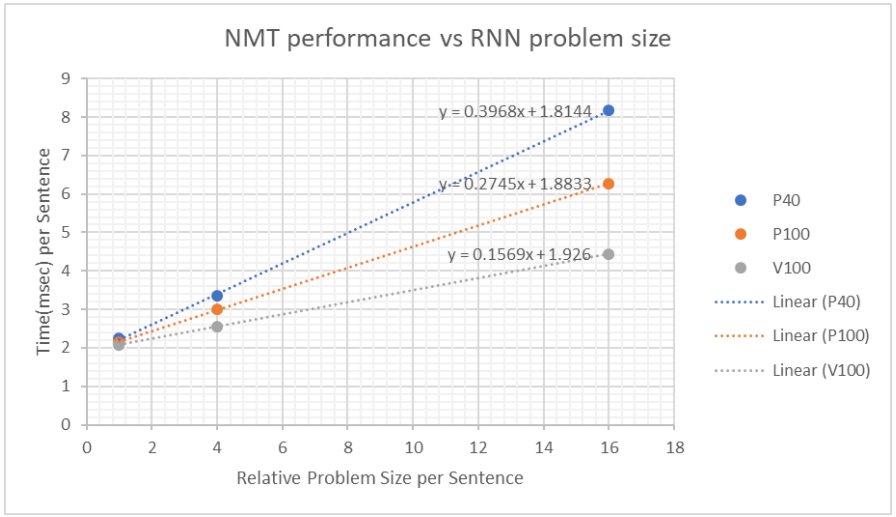
Perf flattens out > batch size 512
P40 very competitive with P100 at smaller batch sizes
P100 does a lot better than P40 at larger batch sizes
P4 is hampered by lower memory capacity (8 GB vs 16 and 24 GB for P40 and P100)
P4 performance pretty good for it's price at low batch sizes (it was designed for inference where small batches are desirable)

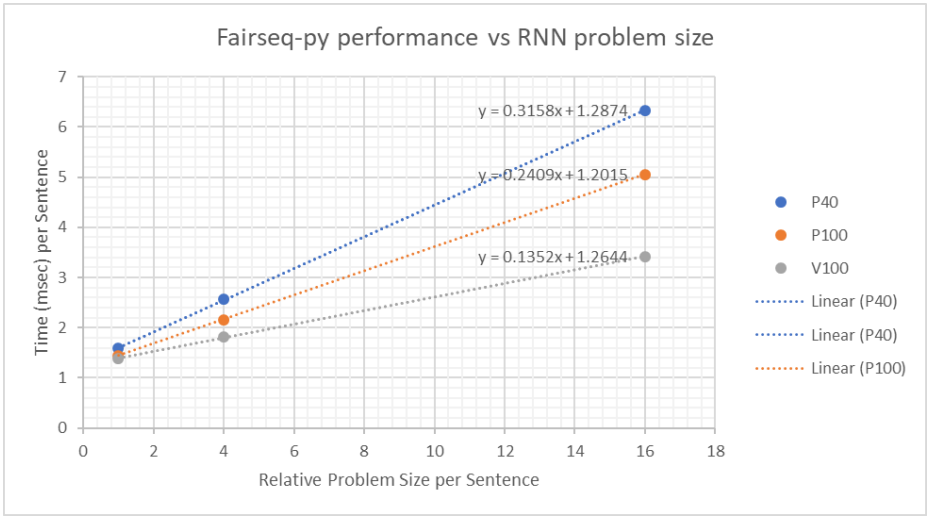


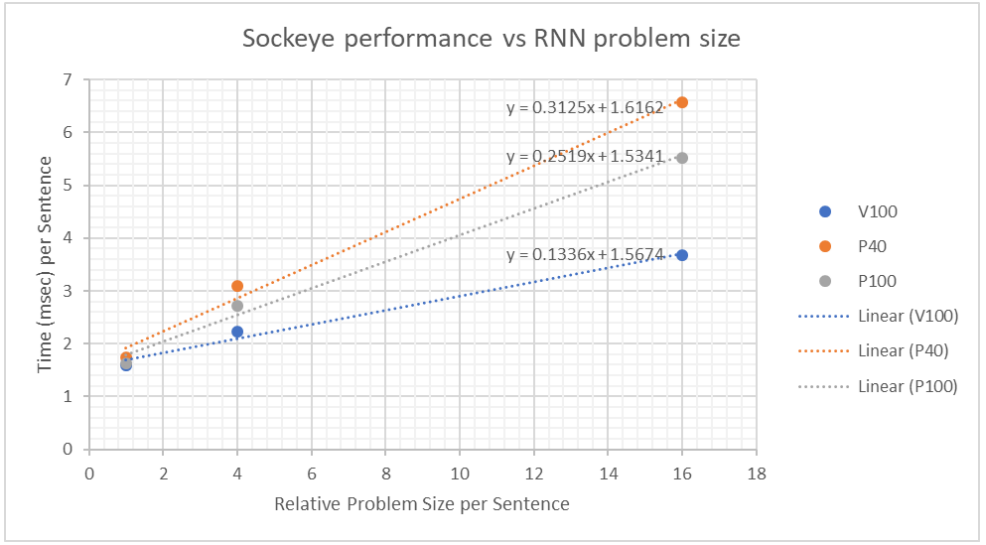




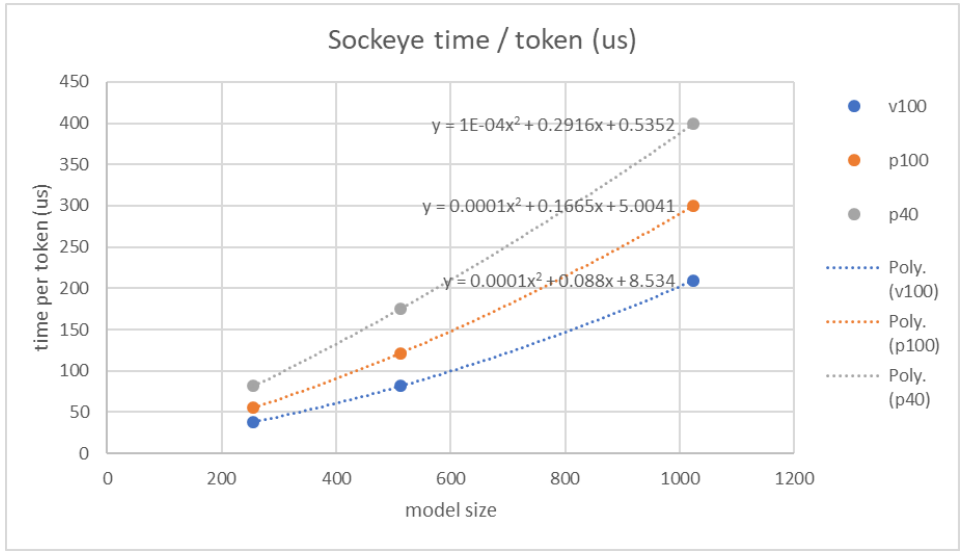
Training speed vs hidden_size^2 (relative units) large baseline independent of GPU capacity

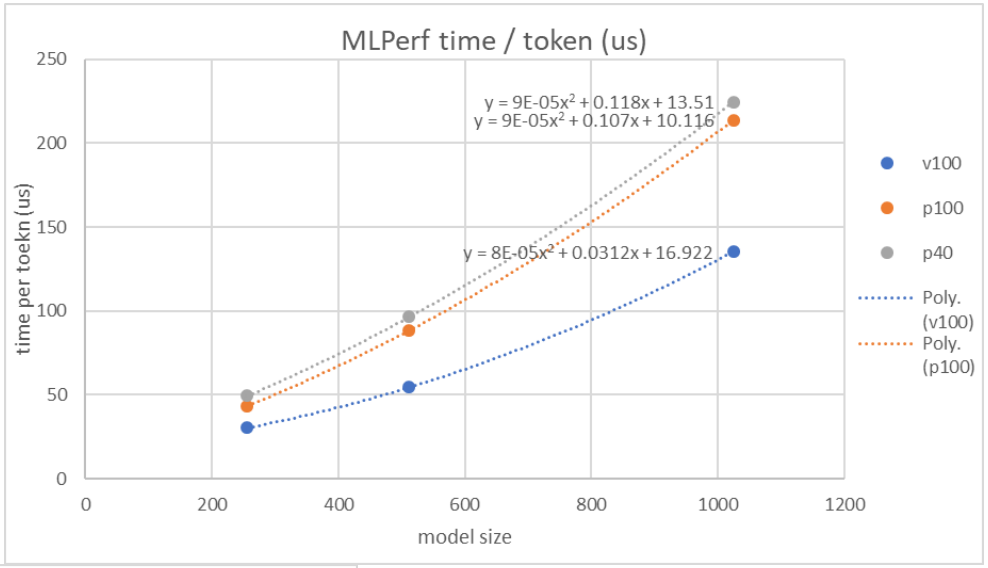


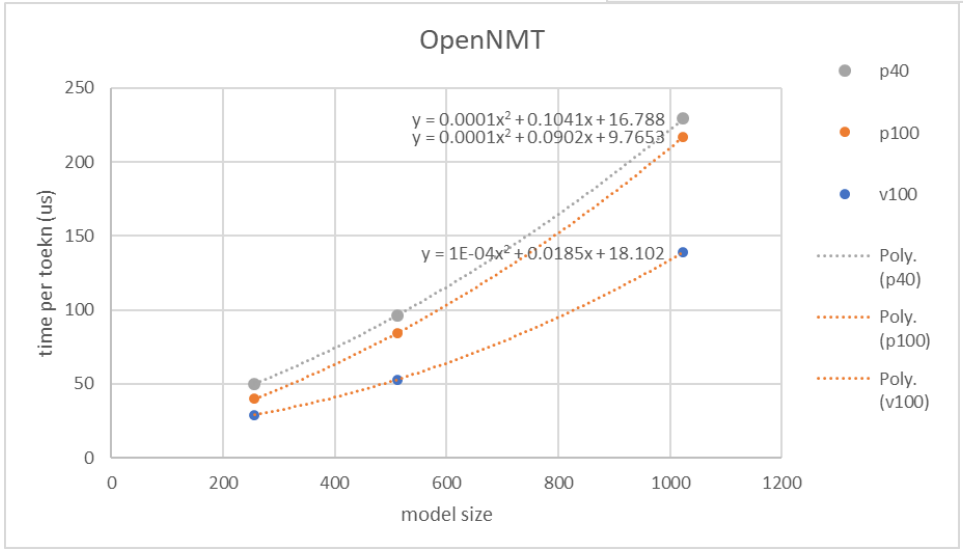




Varying model size in transformers







MLPerf https://mlperf.org/

- Objective multi framework training benchmark run to convergence
- Compare performance/cost of cloud VMs
- Current belief is that equal versions across frameworks can be ported

Usage	implementation
image_classification	resnet50-tensorflow
object_detection	rcnn-caffe2
recommendation	neural filtering-pytorch
reinforcement	minigo-tensorflow
sentiment_analysis	cnn/rnn text categorization-paddle
speech_recognition	deepspeech2-pytorch
translation	transformer-tensorflow

Tools for performance evaluation

- Most machine learning codes are written in python
  - Which invoke compiled framework libraries
  - Which in turn invoke hardware specific math libraries (Cuda, MKL etc)
- As python is an interpreted language, dynamic tracing is required for most analysis.
- There are native python tracers, tracers built into the frameworks in some cases and HW based tools
- HW based tools for CPUs are tied to the performance counters
- HW based tools for GPUs are proprietary (NVProf for Nvidia) and may require binary instrumentation for some measurements

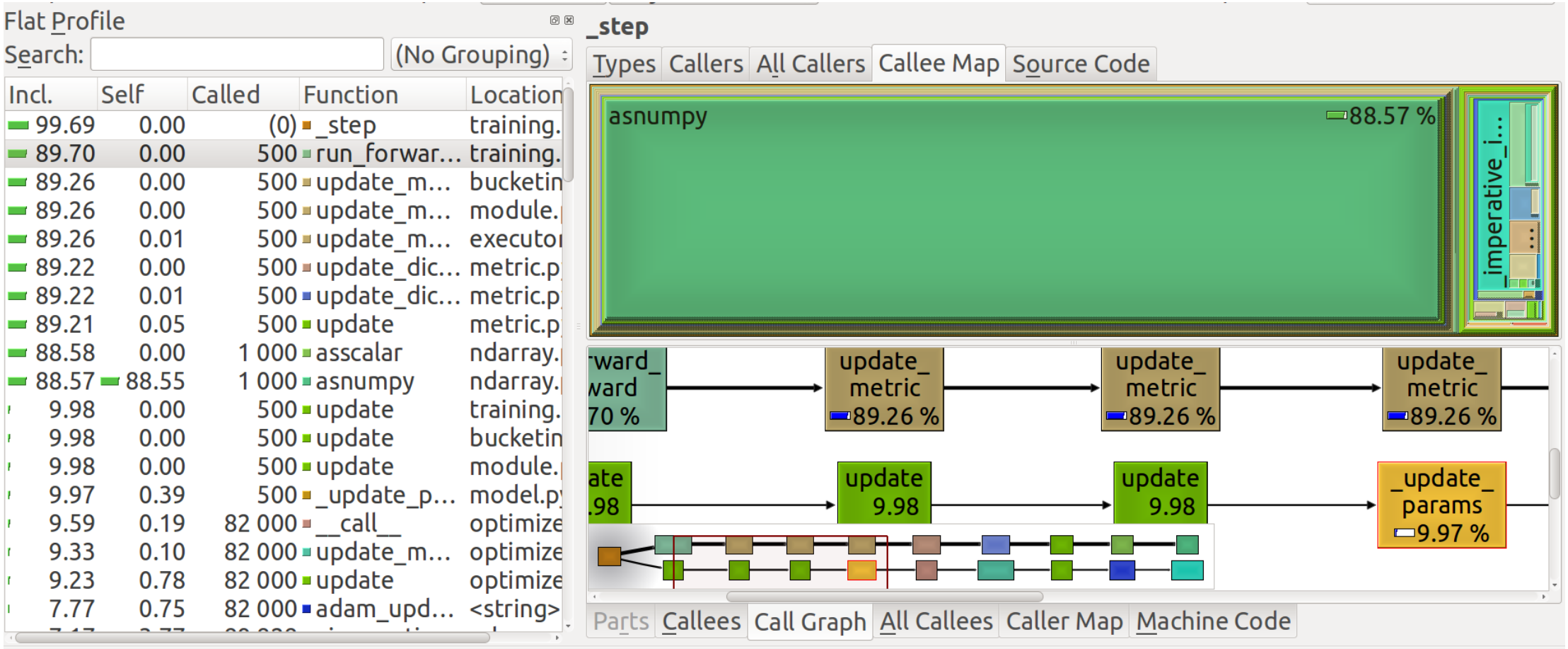
NVProf profiles activity on the GPU only

Slows down code by very large factor (~150X) if things like FP operation counts are collected
Not so bad if only time is collected
Output is CSV, example below is post processed to add some information about the batching and so on

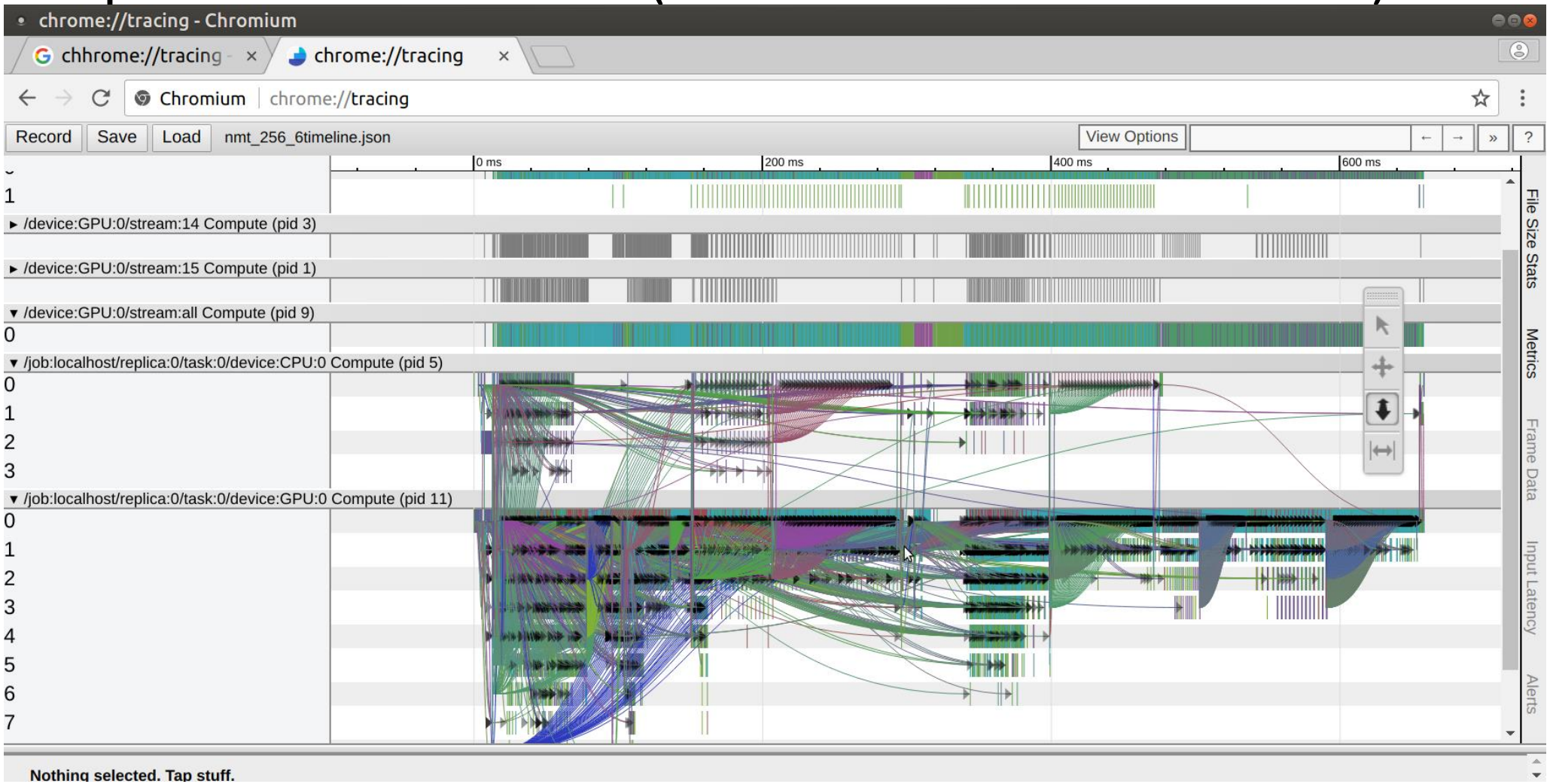
==115269== Profiling application: python tf_cnn_benchmarks.py --model alexnet --forward_only True --local_parameter_device=gpu --num_gpus=1 --batch_size 1 --num_batch 100 --display_every 1000
==115269== Profiling result:
==115269== Metric result:

Device	Kernel	Invocations	Metric Name	Metric Description	Min	Max	Avg	Total
Tesla P4 (0)	void Eigen::internal::EigenMetaKernel<Eigen::TensorEvaluator<Eigen::'	108	flop_count_sp	Floating Point Operations(Single	154,587	154,587	154,587	16,695,396
Tesla P4 (0)	void cudnn::maxwell::gemm::setOutputKernel<float>(int, float*, float)	53	flop_count_sp	Floating Point Operations(Single	0	0	0	0
Tesla P4 (0)	void cudnn::detail::pooling_fw_4d_kernel<float, float, cudnn::detail::m	159	flop_count_sp	Floating Point Operations(Single	9,216	46,656	29,440	4,680,960
Tesla P4 (0)	void Eigen::internal::EigenMetaKernel<Eigen::TensorEvaluator<Eigen::'	54	flop_count_sp	Floating Point Operations(Single	154,587	154,587	154,587	8,347,698
Tesla P4 (0)	void Eigen::internal::EigenMetaKernel<Eigen::TensorEvaluator<Eigen::'	159	flop_count_sp	Floating Point Operations(Single	1,001	4,096	3,064	487,176
Tesla P4 (0)	maxwell_scudnn_128x64_interior_nn	53	flop_count_sp	Floating Point Operations(Single	143,327,232	143,327,232	143,327,232	7,596,343,296
Tesla P4 (0)	void tensorflow::BiasNCHWKernel<float>(int, float const *, float const ,	265	flop_count_sp	Floating Point Operations(Single	43,264	193,600	101,324	26,850,860
Tesla P4 (0)	void cudnn::winograd::generateWinogradTilesKernel<int=0, float, float	159	flop_count_sp	Floating Point Operations(Single	4,276,224	8,552,448	6,176,768	982,106,112
Tesla P4 (0)	maxwell_scudnn_winograd_128x128_tile148n_nt	159	flop_count_sp	Floating Point Operations(Single	157,827,072	314,720,256	227,453,610	36,165,123,990
Tesla P4 (0)	cudnn::maxwell::gemm::computeOffsetsKernel(cudnn::maxwell::gem	107	flop_count_sp	Floating Point Operations(Single	0	0	0	0
Tesla P4 (0)	maxwell_scudnn_128x64_stridedB_splitK_small_nn	53	flop_count_sp	Floating Point Operations(Single	472,743,936	472,743,936	472,743,936	25,055,428,608
Tesla P4 (0)	void Eigen::internal::EigenMetaKernel<Eigen::TensorEvaluator<Eigen::'	371	flop_count_sp	Floating Point Operations(Single	0	0	0	0
Tesla P4 (0)	void gemv2N_kernel_val<float, float, float, int=128, int=8, int=4, int=4, i	53	flop_count_sp	Floating Point Operations(Single	8,529,311	8,529,311	8,529,311	452,053,483
Tesla P4 (0)	void tensorflow::functor::SwapDimension1And2InTensor3Simple<unsi	54	flop_count_sp	Floating Point Operations(Single	0	0	0	0
Tesla P4 (0)	cudnn::maxwell::gemm::computeBOffsetsKernel(cudnn::maxwell::gen	53	flop_count_sp	Floating Point Operations(Single	0	0	0	0
Tesla P4 (0)	void gemv2N_kernel_val<float, float, float, int=128, int=4, int=4, int=4, i	106	flop_count_sp	Floating Point Operations(Single	34,222,080	76,984,320	55,603,200	5,893,939,200
Tesla P4 (0)	void tensorflow::functor::SwapDimension0And2InTensor3Simple<float	266	flop_count_sp	Floating Point Operations(Single	0	0	0	0
				Total flop count				76,202,056,779
				Batch size				1
				Num batches executed				54
				Num warmup batches				0
				Total batches				54
				Total images				54
				Flop per image				1,411,149,200

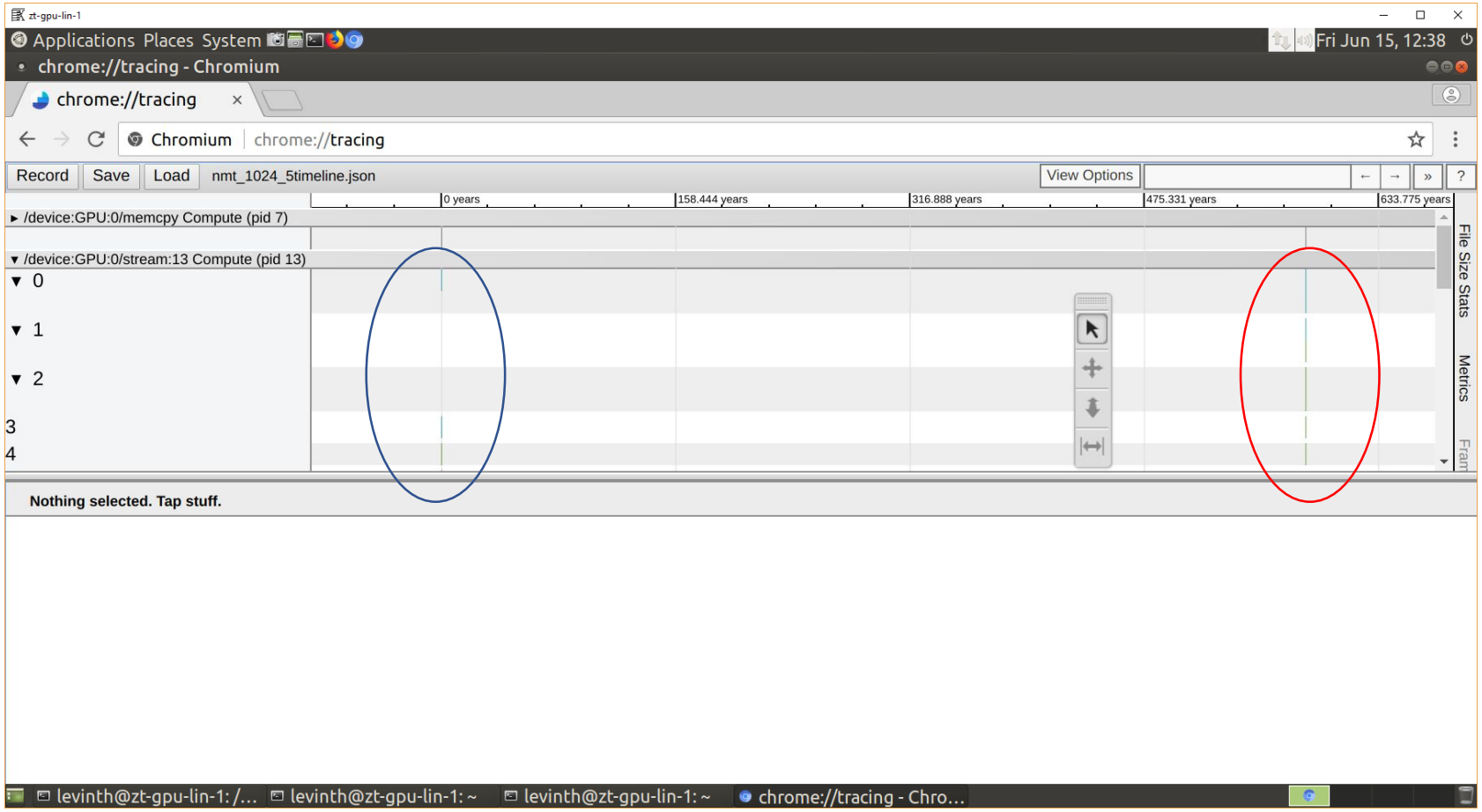
Cprofile only see Python execution



Tracing real ML networks yields a complicated result (Tensorflow timeline)



Hidden size = 1024 minibatch 5
Time stamps of some records went crazy



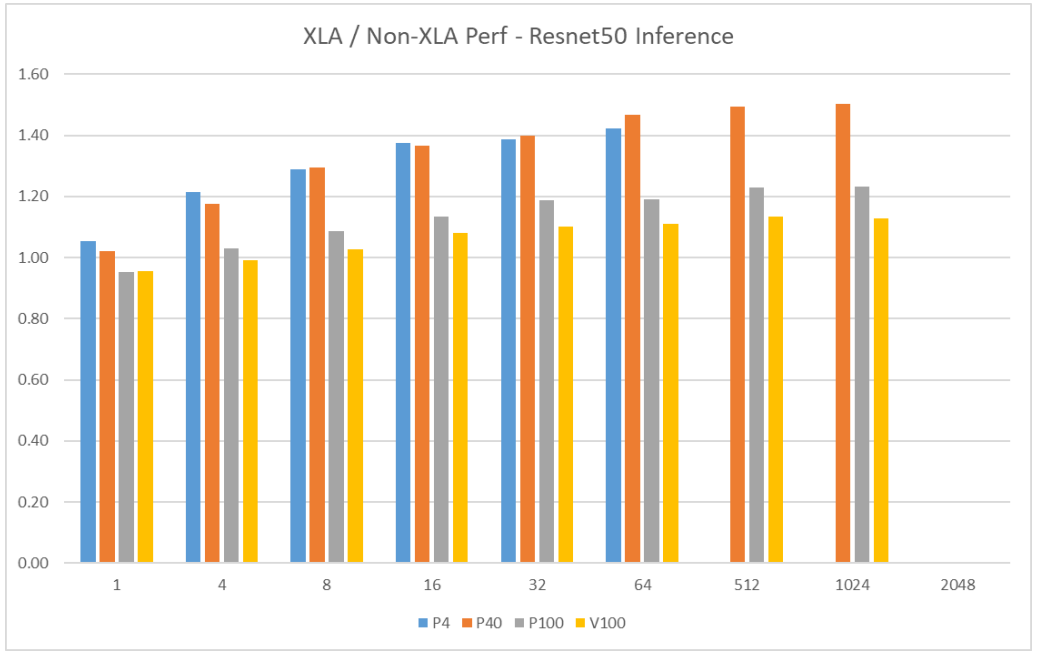
Framework profilers have issues

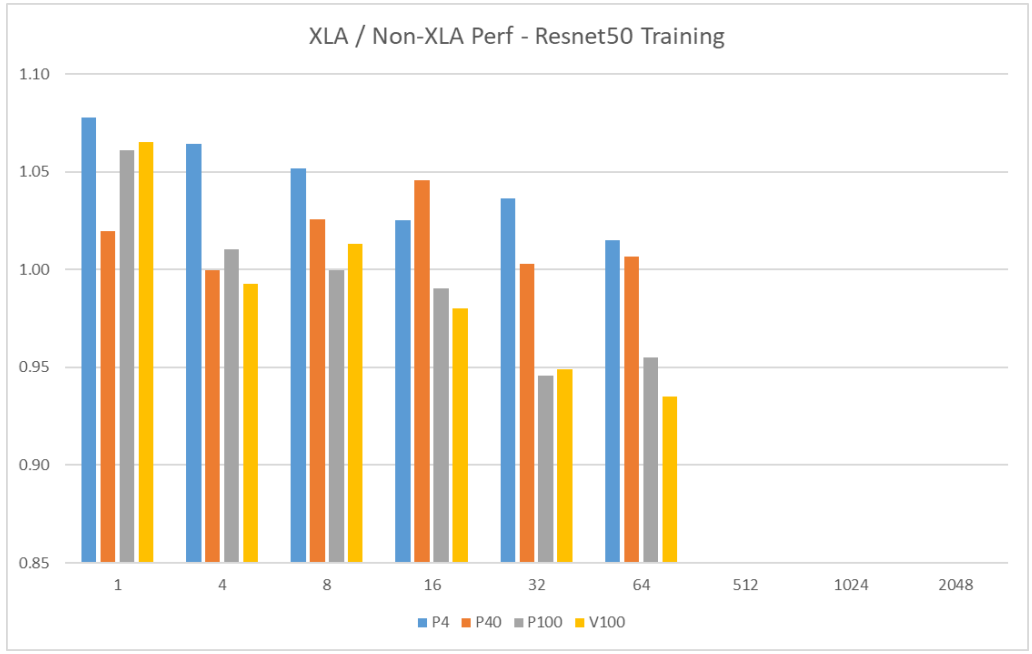
- Not clear TF profiler works
- MxNet profiler has issues with symbols/long names
- Python profilers have issues seeing into compiled libraries
  - ie the frameworks
- HW profilers
  - Nvprof requires binary instrumentation (and 165X slow down) for anything beyond cycles

Intermediate representations

- Intermediate representations for deep neural networks
  - Create a framework independent representation
    - Simplifying multi framework support from HW vendors
  - Enable rational approaches to network calculation optimization
  - Multi layer fusion
    - Ex: conv layer followed by relu layer followed by max pooling layer
    - Combine to a single layer to avoid data movement
    - Multi layer fusion is also done independently from IR ex: Nvidia TensorRT
- XLA and ONNX are currently popular approaches

Impact of XLA on Resnet50 @ fp32





TF R1.7, Cuda 9.1, cudnn 7.1.2

Conclusions

- Understanding performance of machine learning is hard.
- Tools are not good
- Large fractions of time are not in matrix multiplies
  - But we don’t know what is using that time
  - Making HW design improvements a bit difficult