Optimising TensorFlow performance on CPUs

Wolfgang Hayek, Alexander Pletzer, Chris Scott

Science Coding Conference 2019

New Zealand eScience Infrastructure

Overview

1. TensorFlow and why bother with CPUs?
2. Threading and choosing the right package
3. Performance results
4. Summary

TensorFlow and why bother with CPUs?

TensorFlow

- “Open source framework for dataflow and differentiable programming”
- Very popular for machine learning/deep learning
- Graph/Operator architecture enables parallelism
- Some operators implemented using parallel “Eigen”, “MKL-DNN”, or “cuDNN” kernels on CPU and GPU

Why bother?

Deep Learning:

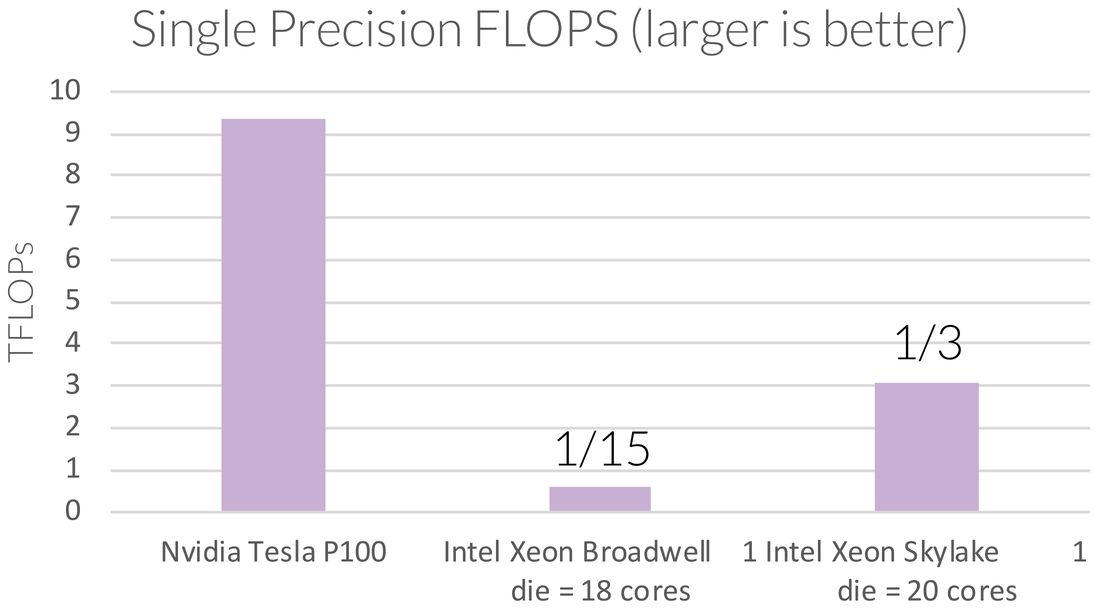
- Very compute intensive (matrix multiplication)
- Modern CPU and GPU hardware well-suited
- Specialised hardware features for Deep Learning and inference available

CPU vs GPU:

- GPUs often faster but not always available
- If I/O is a bottleneck, GPU advantage can be small

Why bother?

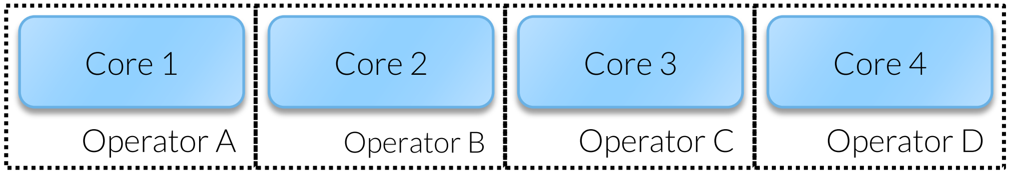
What the datasheets say: GPU card vs 1 CPU die



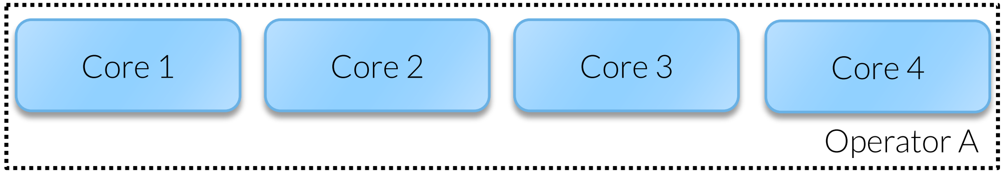
Threading and choosing the right package

Thread Parallelism in TensorFlow on CPUs

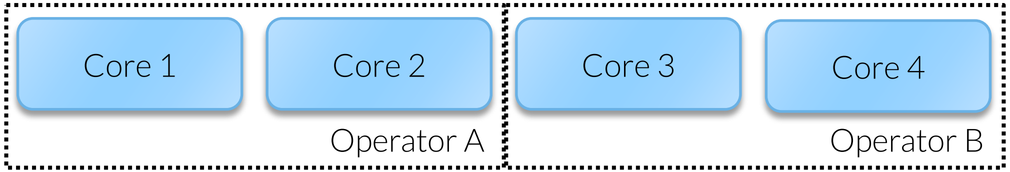
1. Inter-operator – multiple operators in parallel



2. Intra-operator – single operator in parallel



3. Mixed inter-intra-operator parallelism



Allocating Threads

- Best inter/intra split depends on model:
  - How many operators can run in parallel
  - Sufficient work per operator for parallelisation
- Try out different combinations!
- Intel MKL-DNN and Eigen threading:
  `OMP_NUM_THREADS=#intra-operator threads`

Threading

Runtime configuration

- Use thread affinity – pin threads to cores
- Use low thread blocktime with Intel MKL-DNN
- Default in Conda package for TensorFlow (!)
- Use Conda package with “mkl_py<...>” build

Intel recommendation:

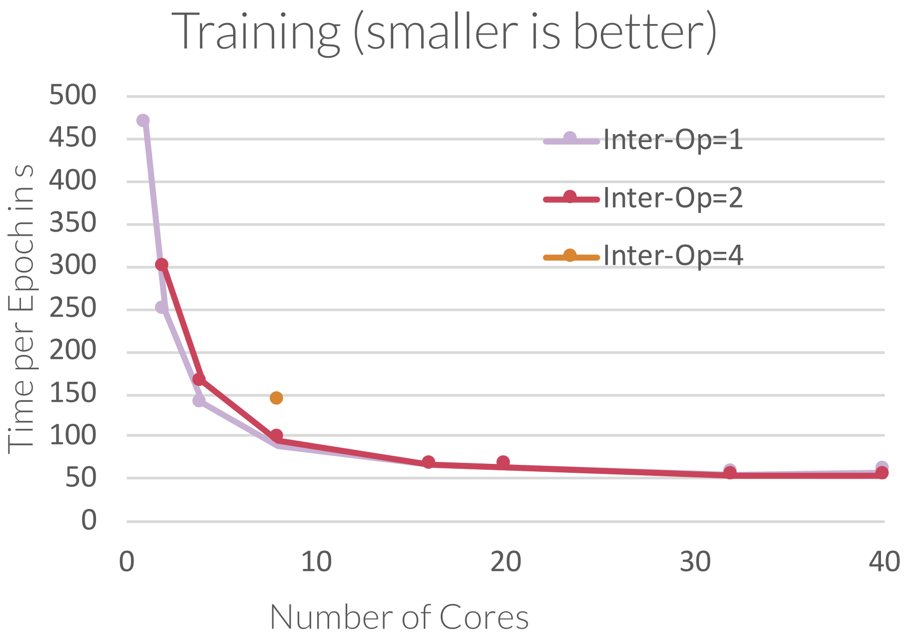
```
KMP_BLOCKTIME=0
KMP_AFFINITY=granularity=fine,compact,0,0
KMP_HW_SUBSET=1T
OMP_NUM_THREADS=intra_op_parallelism_threads
```

Performance results

Performance Results

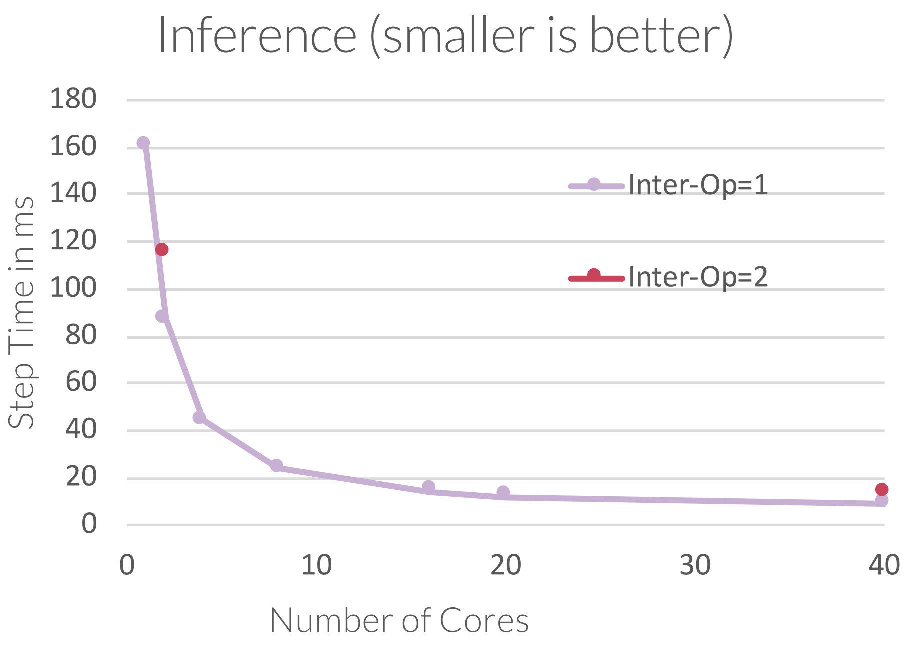
- Classification of video images
- Neural network based on Inception v3
- Work by C. Peat et al. (in publication)
- Test systems:
  1. Nvidia Tesla P100
  2. Intel Xeon Skylake 2.4 GHz x40 node
  3. Intel Xeon Broadwell 2.1 GHz x36 node
- Use optimal threading configuration

Performance Results – Training Scaling



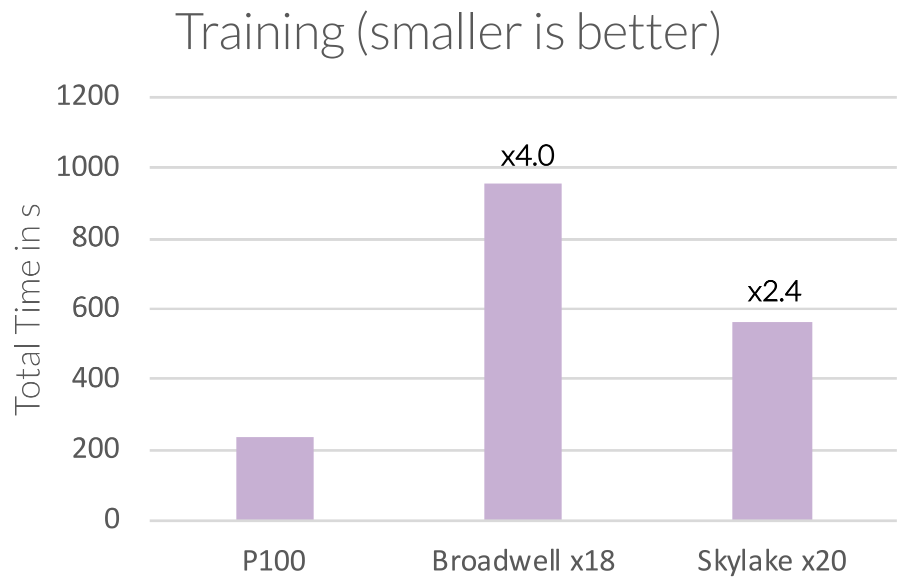
- Skylake node with 40 cores
- Total number of cores = Inter-Op x Intra-Op
- Batch Size 32

Performance Results – Inference Scaling



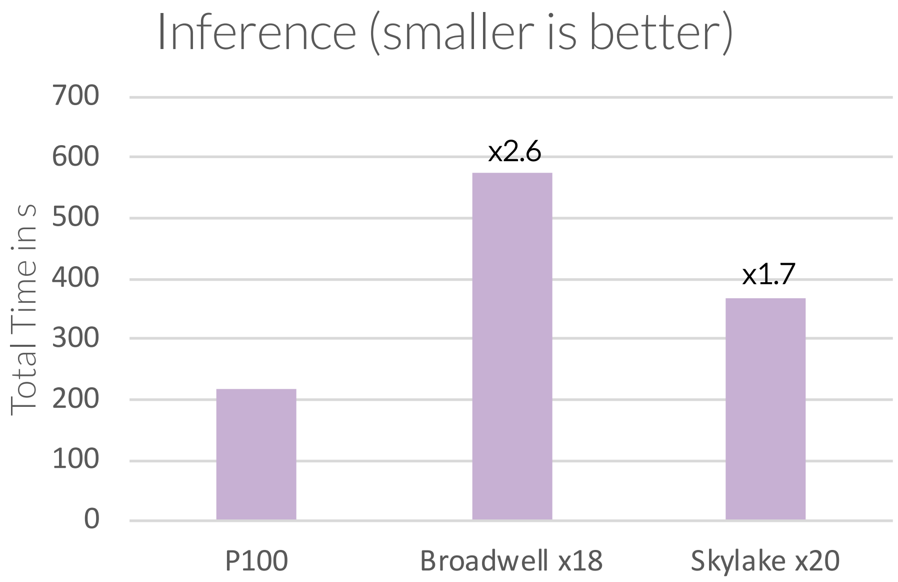
- Batch Size 128

Performance Results – Overall Training Time



- Batch Size 32 – not always ideal for performance
- Use all cores in a CPU die (1/2 node)
- CPU slower, but not order of magnitude

Performance Results – Overall Inference Time



- Batch Size 100
- CPU can be competitive!

Summary

Summary

- CPU can deliver competitive performance
- Explore MPI+Threading for further improvement
- Will depend on model and workflow
- Could run training on GPU, inference on CPU
- Check out mlperf.org!

Let us know if you would like help: support@nesi.org.nz

Thank you!