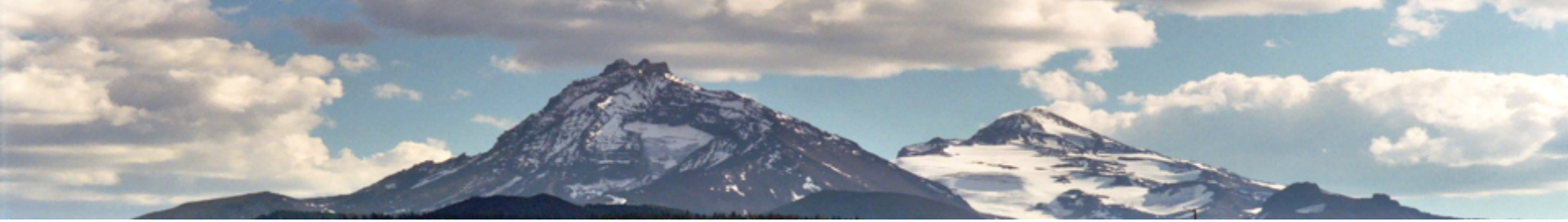


Performance Analysis of CUDA Deep Learning Networks using TAU

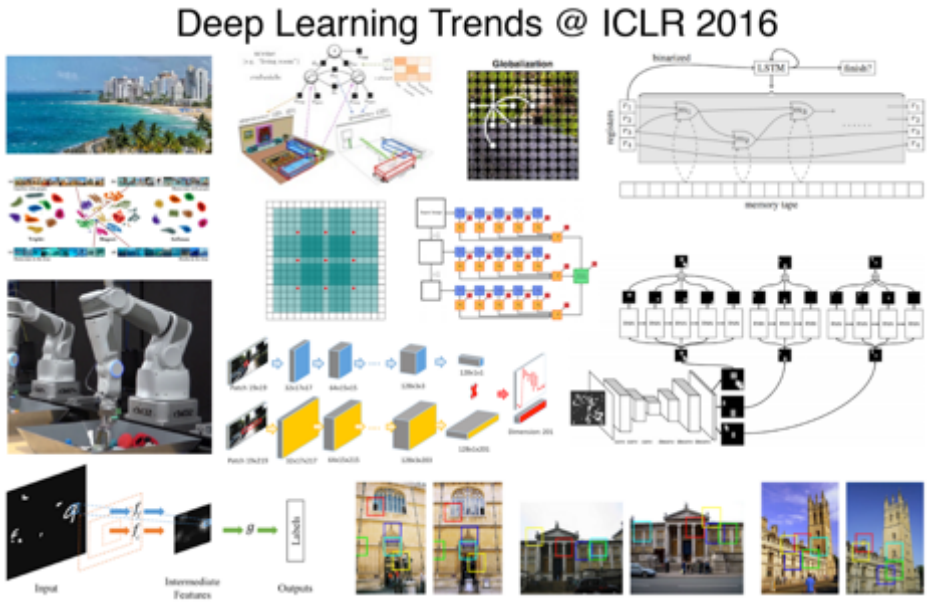
Allen D. Malony, Robert Lim,
Sameer Shende, Boyana Norris

Department of Computer and Information Science
Oregon Advanced Computing Institute for Science and Society

University of Oregon

Motivation

- Tremendous growth in the application of deep neural network (DNN) to deep learning (DL) applications
  - Image recognition:
    - Facebook photo tagging
  - Text recognition:
    - Google Translate
  - Speech recognition:
    - Baidu Deep Voice, ...



facebook Google Baidu Research

- DNN libraries and DL frameworks for programming
  - CUDA Deep Neural Network (cuDNN) library
  - Frameworks built around cuDNN for DL application creation
    - Torch, TensorFlow, Caffe/Caffe2, Neon, MXnet, theano, DL4j, …

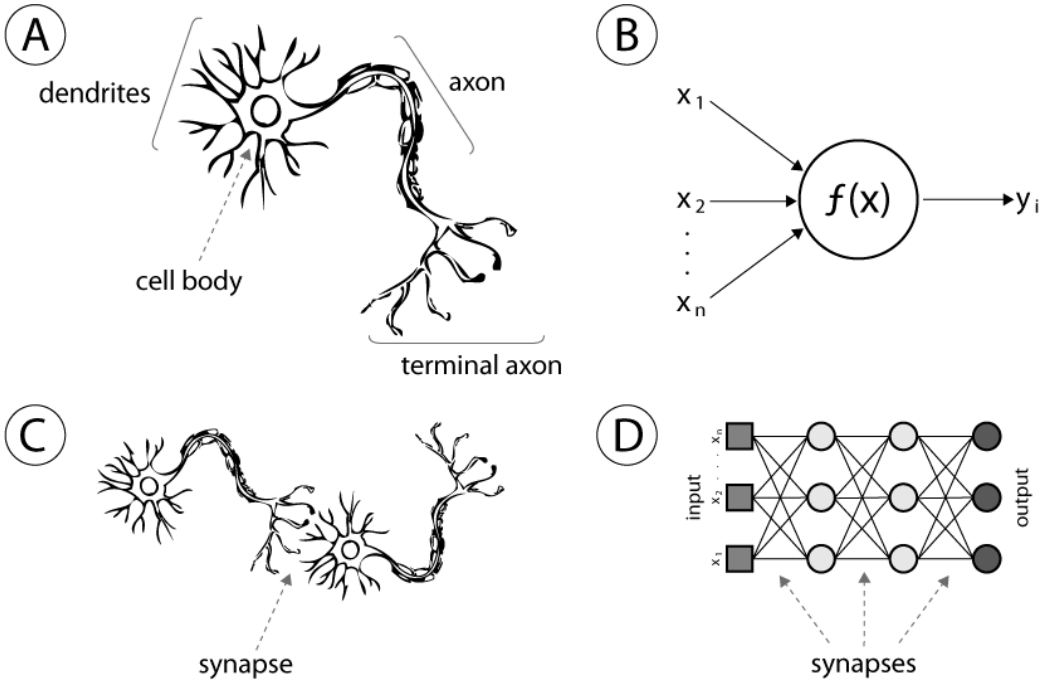
Motivation (2)

- DNN/DL applications are complex (function, execution)
- Build DNN/DL development and evaluation ecosystem
- Provide support for performance measurement/analysis
  - Address concerns for users wanting to tune performance
  - Optimize sub-par configurations or target execution bugs
  - Understand performance relative to user's high-level problem
    - execution time spent per layer, accuracy of trained results, …
- Target rich platforms for integration (libraries, frameworks)
- Integrate tools in CUDA development environment
  - Lack ability to report intricacies of cuDNN activity
  - Report behavior of DNN features at routine level
  - Confirm and optimize model/network configuration performance

Outline of Talk

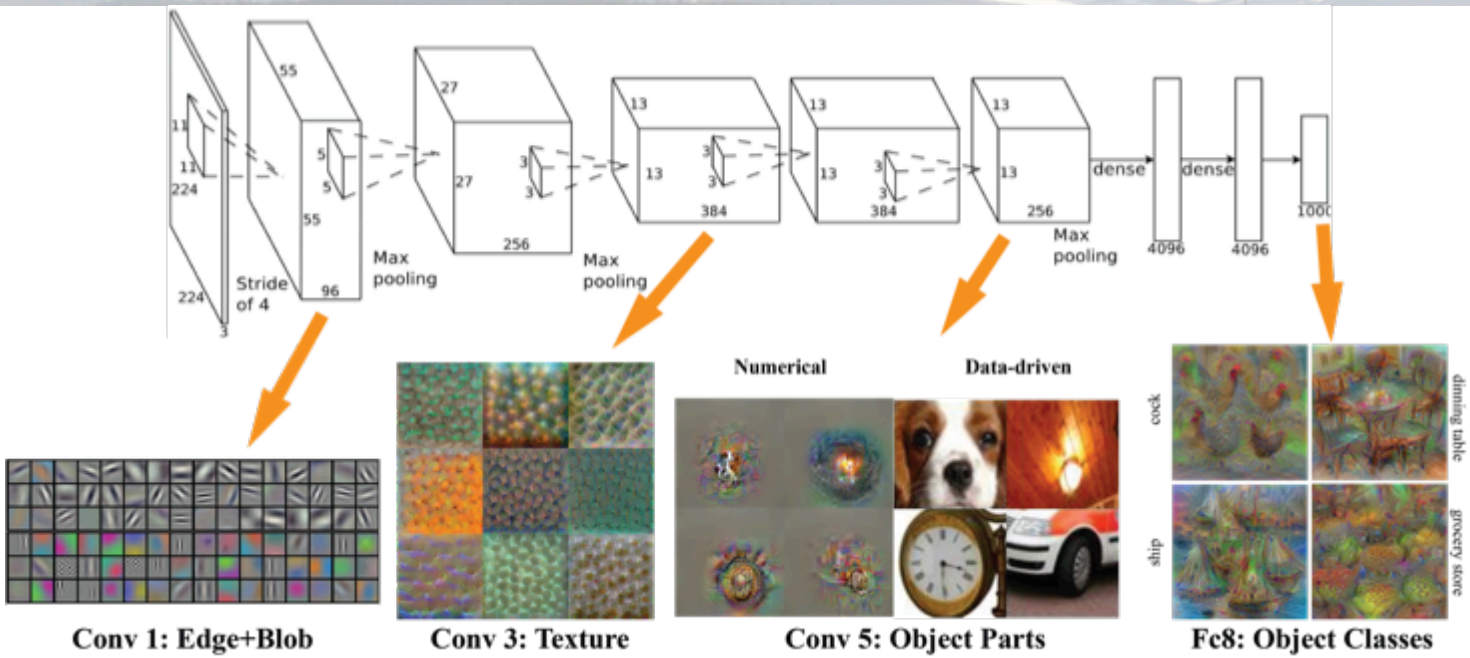
- DNN libraries and DL frameworks
- Challenges in DNN/DL performance and analysis
  - Performance problems of concern
  - Application of CUDA performance tools to DNN/DL
- TAU Performance System
  - Overview
  - Support for CUDA performance analysis
- TAU prototype for DNN/DL performance analysis
  - Test know DNN benchmarks with different DL frameworks
  - Run on a variety of GPUs
  - See what DNN characteristics can be revealed

Deep Learning and Neural Networks



- Uses brain-inspired computing to process fine-grained tasks
- Decomposes problem into domain specific tasks
  - Tasks solve specific problem
  - Results collectively provide high level answers

Convolutional Neural Networks (CNN) Basics



- Convolution layer
  - Feature detector that learns to filter out unneeded information
    - apply a convolutional kernel to the input
- Pooling layers
  - Compute metrics on particular feature over region of input data
  - Also detects objects in unusual places, reduces memory size

Deep Neural Network (DNN) Workflow

- DNN frameworks implement a workflow made of a sequence of standard, common stages
  - Programming can be imperative or declarative

1) Target a backend (CPU or GPU, or both)

2) Load data

3) Specify model architecture
   - Create model by providing list of layers
   - Layers with weights:
     - provide function to initialize weights prior to training
     - layers (linear, convolution, pooling)
     - activations (RELU, softmax tanh)
     - initializers (constant, uniform, gaussian)

Deep Neural Network (DNN) Workflow (2)

4) Train model

- Provide training data (as an iterator), cost function and optimization algorithm for updating model's weights
- Learning schedule:
  - modify learning rate over training time
  - datasets, costs and metrics
  - optimizers (SGD, adagrad, adam)
  - learning schedules

5) Evaluate

- Evaluate trained model on validation dataset and metrics
- Models, costs (cross entropy, SSE), metrics (LogLoss, PrecisionRecall)

NVIDIA CUDA DNN Library (cuDNN)

- GPU-accelerated library for DNN
- Provides highly-tuned implementations
  - Standard routines: forward/backward convolution, pooling, normalization, activation layers
  - Part of NVIDIA deep learning SDK
- Deep learning researchers and framework developers worldwide rely on cuDNN for acceleration
  - Focus on training neural networks and developing software applications rather than tuning low-level GPU performance
- cuDNN accelerates widely-used DL frameworks

Key Features of cuDNN

- Forward/backward paths for many common layer types
  - Pooling, LRN, LCN, and batch normalization
  - ReLU, Sigmoid, softmax, and Tanh
- Forward and backward convolution routines
  - Cross-correlation
  - Designed for convolutional neural networks (CNN)
- Recurrent Neural Networks (RNN) and Persistent RNN
  - LSTM (long short-term memory)
  - GRU (gated recurrrent unit)
- 4d tensors
  - Arbitrary dimension ordering, striding, sub-regions
- Tensor transformation functions
- Context-based API for easy multithreading

Deep Learning (DL) Frameworks

- DL frameworks effectively implement DL workflows
- Large variety of DL frameworks

https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software

Software	Creator	Software license[a]	Open source	Platform	Written in	Interface	OpenMP support	OpenCL support	CUDA support	Automatic differentiation[1]	Has pretrained models	Recurrent nets	Convolutional nets	RBM/DBNs	Parallel execution (multi node)
Apache Singa	Apache Incubator	Apache 2.0	Yes	Linux, Mac OS X, Windows	C++	Python, C++, Java	No	Yes	Yes	?	Yes	Yes	Yes	Yes	Yes
Wolfram Mathematica	Wolfram Research	Proprietary	No	Windows, Mac OS X, Linux, Cloud computing	C++	Command line, Java, C++	No	Yes	Yes	Yes	Yes[50]	Yes	Yes	Yes	Yes
Caffe	Berkeley Vision and Learning Center	BSD license	Yes	Linux, Mac OS X, Windows[2]	C++	Python, MATLAB	Yes	Under development[3]	Yes	Yes	Yes[4]	Yes	Yes	No	?
Theano	Université de Montréal	BSD license	Yes	Cross-platform	Python	Python	Yes	Under development[34]	Yes	Yes[35][36]	Through Lasagne's model zoo[37]	Yes	Yes	Yes	Yes[38]
Keras	François Chollet	MIT license	Yes	Linux, Mac OS X, Windows	Python	Python	Only if using Theano as backend	Under development for the Theano backend (and on roadmap for the TensorFlow backend)	Yes	Yes	Yes[9]	Yes	Yes	Yes	Yes[10]
Torch	Ronan Collobert, Koray Kavukcuoglu, Clement Farabet	BSD license	Yes	Linux, Mac OS X, Windows,[39] Android,[40] iOS	C, Lua	Lua, LuaJIT,[41] C, utility library for C++/OpenCL[42]	Yes	Third party implementations[43][44]	Yes[45][46]	Through Twitter's Autograd[47]	Yes[48]	Yes	Yes	Yes	Yes[49]
Deeplearning4j	Skymind engineering team; Deeplearning4j community; originally Adam Gibson	Apache 2.0	Yes	Linux, Mac OS X, Windows, Android (Cross-platform)	java	Java, Scala, Clojure, Python (Keras)	Yes	On roadmap[5]	Yes[6]	Computational Graph	Yes[7]	Yes	Yes	Yes	Yes[8]
TensorFlow	Google Brain team	Apache 2.0	Yes	Linux, Mac OS X, Windows[29]	C++, Python	Python, C/C++, Java, Go	No	On roadmap[30][31]	Yes	Yes[32]	Yes[33]	Yes	Yes	Yes	Yes
MXNet	Distributed (Deep) Machine Learning Community	Apache 2.0	Yes	Linux, Mac OS X, Windows,[21][22] AWS, Android,[23] iOS, JavaScript[24]	Small C++ core library	C++, Python, Julia, Matlab, JavaScript, Go, R, Scala, Perl	Yes	On roadmap[25]	Yes	Yes[26]	Yes[27]	Yes	Yes	Yes	Yes[28]
Dlib	Davis King	Boost Software License	Yes	Cross-Platform	C++	C++	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes
Microsoft Cognitive Toolkit	Microsoft Research	MIT license[11]	Yes	Windows, Linux[12] (OSX via Docker on roadmap)	C++	Python, C++, Command line,[13] BrainScript[14] (.NET on roadmap[15])	Yes[16]	No	Yes	Yes	Yes[17]	Yes[18]	Yes[18]	No[19]	Yes[20]
Neural Designer	Artelnics	Proprietary	No	Linux, Mac OS X, Windows	C++	Graphical user interface	Yes	No	No	?	?	No	No	No	?
OpenNN	Artelnics	GNU LGPL	Yes	Cross-platform	C++	C++	Yes	No	No	?	?	No	No	No	?

Torch

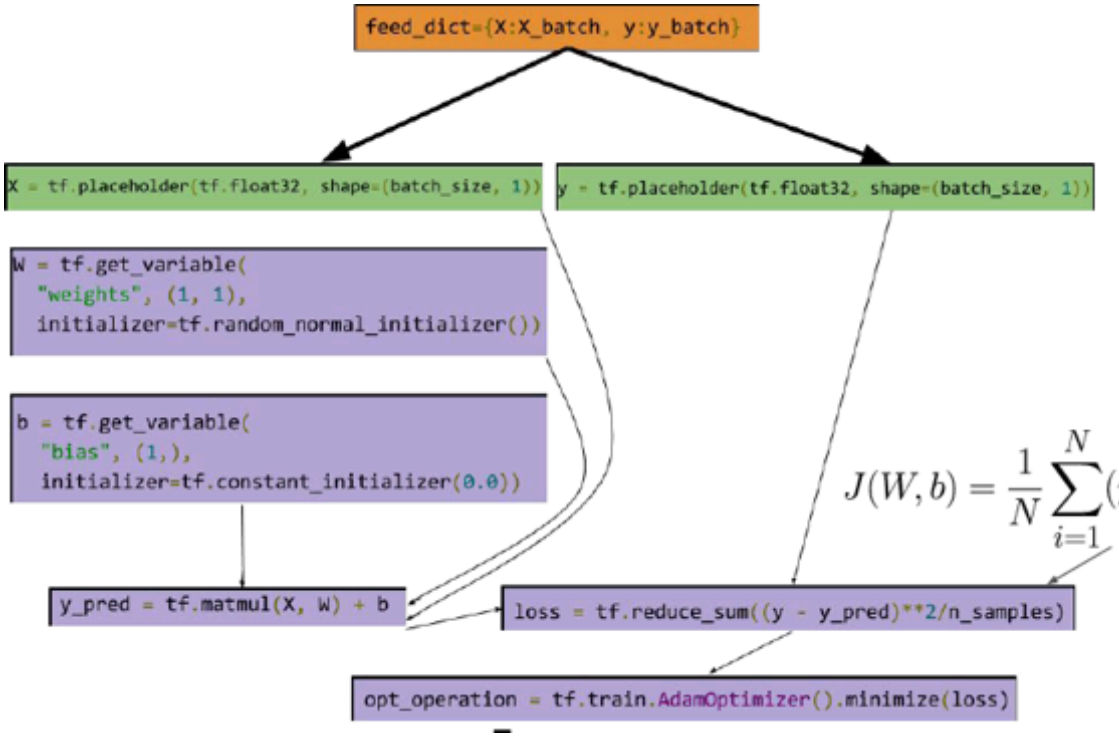
- Torch
  - Popular scientific framework
    - LuaJIT and Python (PyTorch) flavors
    - Has its roots with Facebook AI Research (circa 2000)
  - Best performer of DNN libraries on convnet benchmarks
- Neural network (nn) package
  - *Module*: abstract class inherited by Module
  - *Containers*: Sequential, Parallel, Concat
  - *Transfer functions*: Tanh, Sigmoid
  - *Simple layer*: Linear, Mean, Max, Reshape
  - *Table layers*: SplitTable, ConcatTable, JoinTable
  - *Convolution layers*: Temporal, Spatial, Volumetric
  - *Criterion*

TensorFlow

- TensorFlow
  - Deep learning library from Google
  - Open source (acquired under DeepMind project)
  - Primitives for defining functions on tensors and automatically computing derivatives
- Numerical computation using data flow graphs
  - Nodes represent mathematical operations
  - Graph edges represent the multidimensional data arrays (tensors) communicated between them
  - Flexible architecture supports CPU and GPU execution
  - Executes deep learning routines on GPU (cuDNN)

TensorFlow Computation Graph

- Placeholders - input nodes in TF computational graph
- Feed dictionaries - how users set values for placeholder (or other) variables when running a computation



TF graph for linear regression

$$J(W,b) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (Wx_i + b))^2$$

Neon

- neon is an open source Python-based language and a set of libraries for developing deep learning models
  - Acquired by Intel
  - Claims to be very fast

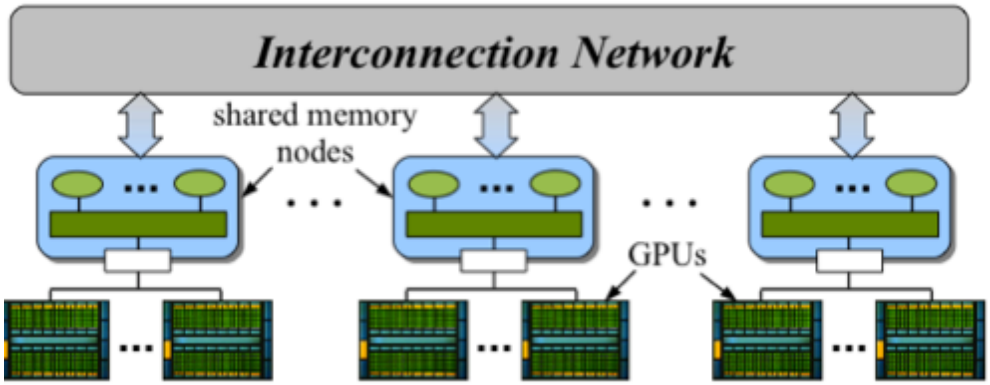
- Python-like syntax
- Object-oriented implementations of all the deep learning components, including layers, learning rules, activations, optimizers, initializers, and costs functions
- All common deep learning models, including convnets, MLPs, RNNs, LSTMs and autoencoders
- Create novel algorithms using linear algebra, auto-differentiation, and other advanced capabilities with a numpy-like syntax

Profiling DNN / DL Applications

- Performance of DNN/DL applications is important
  - Considerable time is spent in training and inference
  - How can these be sped up?
  - Understand performance inefficiencies
  - Understand opportunities for improvement
- To date, metrics reported for DNN applications include:
  - Classification accuracy
  - Time spent per layer (forward, backward)
- Currently lacking the ability to report operations executed, memory footprint, system aspects, …
- Performance measurements of interest for developers:
  - Input resolution, ops per layer or module, memory access patterns, # hidden neurons, …
  - Whether DNN training is making significant progress

Heterogeneous HPC and Performance Tools

- Heterogeneous systems drive HPC performance
  - Multi-CPU, multicore shared memory nodes
  - Manycore accelerators with high-BW I/O

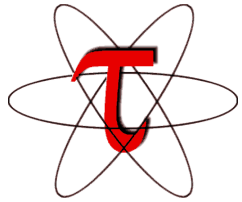


- Heterogeneous software development technology important to deliver on performance potential
  - More sophisticated parallel programming environments
  - Integrated development and performance tools
    - support heterogeneous performance model and perspectives
  - HPC tools research on heterogeneous GPU computing

Heterogeneous + GPU Performance Analysis

- Heterogeneous HPC concerns node-level and cluster-wide performance issues
  - Maintain high concurrency and keep accelerators busy
  - Balancing of load across nodes
  - Reducing overheads and increasing efficiency
- Need to achieve high GPU performance as well
  - Multiple parameters are involved
    - threads, blocks, registers, shared memory, …
  - Data parallel programming concerns
    - work size, locality, branch divergence, ...
  - Need better support for analysis and optimization

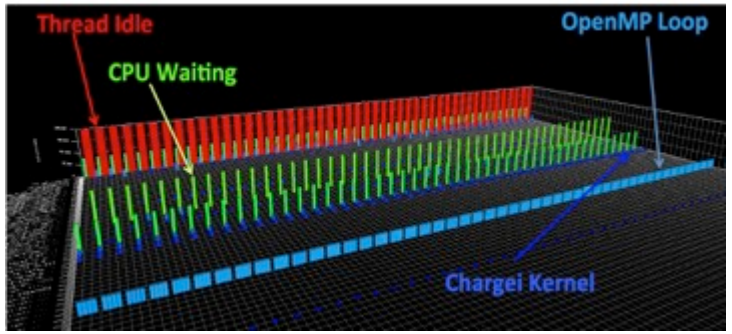
TAU Performance System® (http://tau.uoregon.edu)

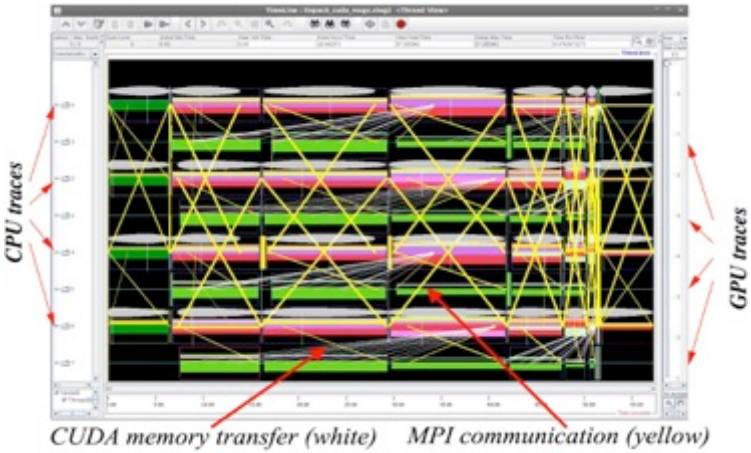


- Performance problem solving framework
  - Integrated, scalable, flexible, portable
  - Target all parallel programming / execution paradigms
- Integrated performance toolkit (open source)
  - Multi-level performance instrumentation
  - Flexible and configurable performance measurement
  - Widely-ported performance profiling / tracing system
  - Performance data management and data mining
  - Open source (BSD-style license)
- Used in HPC software, systems, applications

Integrated Heterogeneous Support in TAU

- State-of-the-art comprehensive HPC performance analysis
  - Multicore, node-level, communication, manycore accelerator
- GPU performance analysis
  - GPU measurement (CUPTI enabled)
    - CUDA library wrapping/callback
    - timing, counters, sampling, transfer
  - Languages
    - CUDA, OpenCL, OpenACC
    - OpenMP 4.x with offloading
  - Integrated profiling and tracing
  - Ports to Linux x86_64, CrayCNL, ARM64, and Power 8 Linux
- Static analysis of GPU kernel
  - Instruction mix, control flow, memory, occupancy, time, …
- Autotuning with static+dynamic analysis and modeling





Configuring TAU with DNN / DL Libraries

- TAU currently supports the following libraries
  - TensorFlow, Nervana Neon, Torch (PyTorch), Theano
  - Near alpha: cuDNN, Torch (Lua)
  - Beta: DL4J
- Example config command:

  ./configure -boost=/home/roblim1/apps/boost_1_61_0 -pythoninc=/home/roblim1/anaconda2/include/python2.7 -pythonlib=/home/roblim1/anaconda2/lib -cc=gcc -cuda=/cm/extra/apps/cuda80/toolkit/8.0.27 -pdt=/home/roblim1/repos/pdtoolkit-3.20 -bfd=download && make install
- TAU makefiles allow multiple installations to coexist based on config settings (MPI, OpenMP, CUDA, etc.)

  export TAU_MAKEFILE='/home/roblim1/repos/tau2/x86_64/lib/Makefile.tau-papi-python-cupti-pdt'

PyTau and tau_python

- TAU provides several ways of measuring performance for Python-based applications
  - `tau_python` source code wrapping
  - `pytau` instrumentation
- Python source code wrapping
  - Parses the Python source code
  - Wraps each routine with TAU Python instrumentation
- `tau_python` example:

```
tau_python -T cupti,serial -cupti mnist_mlp.py
```

PyTau and tau_python (2)

- TAU Python measurement API (`pytau`)
  - Calls the underlying TAU performance measurement
- `pytau` example:

```
from neon.models import Model
from neon.optimizers import GradientDescentMomentum
from neon.transforms import Rectlin, Logistic, CrossEntropyBinary, Misclassification
import pytau
…
x = pytau.profileTimer("MLP Fit")
pytau.start(x)
mlp.fit(train_set, optimizer=optimizer, num_epochs=args.epochs, cost=cost,
        callbacks=callbacks)
error_rate = mlp.eval(valid_set, metric=Misclassification())
pytau.stop(x)
pytau.dumpDb()
…
```

GPU Hardware Counters

- Configure TAU to access hardware counters
  - PAPI for CPU and CUPTI for GPU
- tau_cupti_avail: lists available GPU counters

```
...
CUDA.Tesla_P100-PCIE-16GB.domain_d.active_warps        active_warps
CUDA.Tesla_P100-PCIE-16GB.domain_d.atom_count          atom count
CUDA.Tesla_P100-PCIE-16GB.domain_d.branch              branch
CUDA.Tesla_P100-PCIE-16GB.domain_d.divergent_branch    divergent branch
...
```

- Calculate derived metrics, such as:
  - Instructions per cycle, global store per instruction
- Enable counters “instructions executed” + “active cycles”

```
export TAU_METRICS='CUDA.Tesla_P100-PCIE-16GB.domain_d.inst_executed:
CUDA.Tesla_P100-PCIE-16GB.domain_q.active_cycles'
```

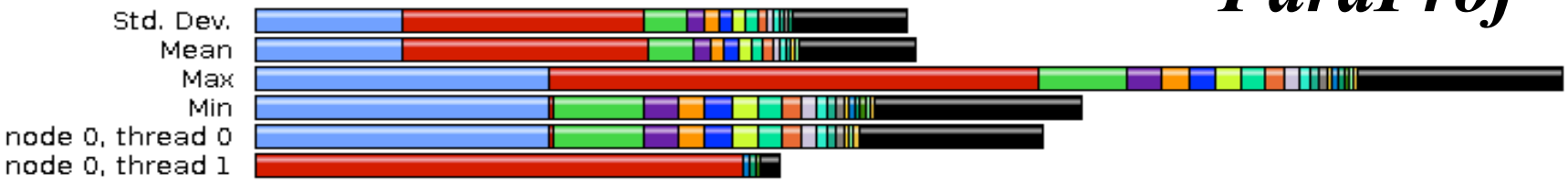
Profile Example with Neon and Alexnet

TAU: ParaProf: /wopr/users/roblim1/outfiles/convnet/neon/alexnet/p100

File Options Windows Help

Metric: TAUGPU_TIME
Value: Exclusive

ParaProf

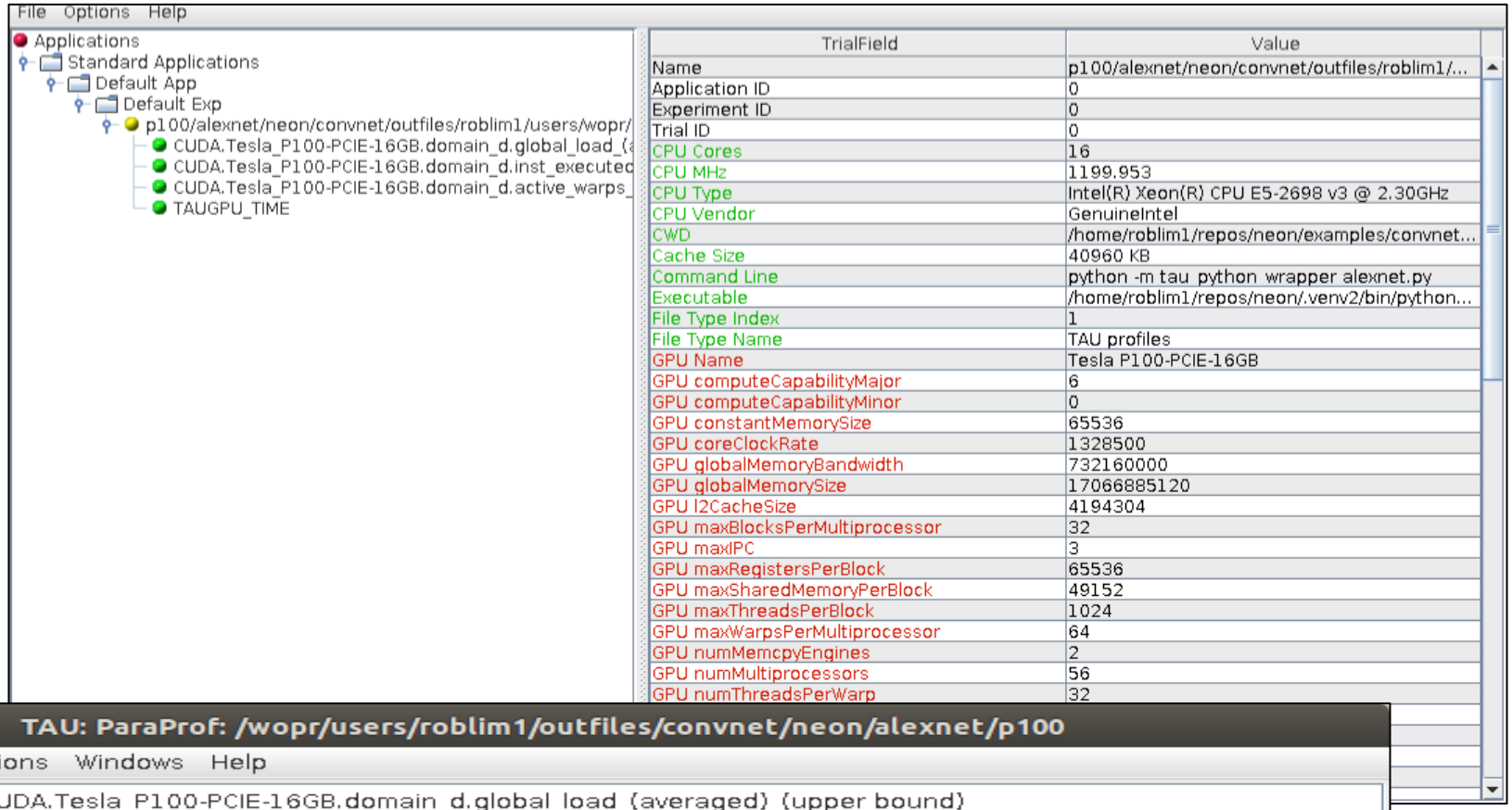


pprof

```
NODE 0;CONTEXT 0;THREAD 1:
---------------------------------------------------------------------------------------
%Time    Exclusive    Inclusive       #Call      #Subrs Count/Call Name
              counts total counts
---------------------------------------------------------------------------------------
100.0    3.451E+05    3.451E+05           5           0      69012 copy_transpose
100.0    3.451E+05    3.451E+05           5           0      69012 kernel_onehot
 58.3    2.013E+05    2.013E+05           1           0     201280 prepare
 50.0    1.725E+05    1.725E+05          10           0      17253 sconv_direct_fprop_64x128
  0.0            0            0           1         392          0 .TAU application
  0.0            0            0           1           0          0 Memory copy Device to Host
  0.0            0            0          14           0          0 Memory copy Host to Device
  0.0            0            0          15           0          0 bprop_filter_4x4_3x3
  0.0            0            0           5           0          0 filter_dimshuffle
  0.0            0            0          15           0          0 fprop_filter_4x4_3x3
  0.0            0            0          40           0          0 kernel_div_mul_mul_add_mul_add
  0.0            0            0          35           0          0 kernel_gt_lt_mul_add_mul
  0.0            0            0           5           0          0 kernel_max_sub_exp_sum_div_sub_exp_mul
  0.0            0            0          35           0          0 kernel_maximum_minimum_mul_add
  0.0            0            0           5           0          0 kernel_mul
  0.0            0            0           5           0          0 kernel_sub_mul
  0.0            0            0           5           0          0 sconv_direct_bprop_64x128
  0.0            0            0          10           0          0 sconv_direct_updat_64x32
  0.0            0            0          15           0          0 sconv_winograd_3x3_4x4_32x32
  0.0            0            0          30           0          0 sconv_winograd_4x4_3x3_32x32_X_Q13_N128
  0.0            0            0          15           0          0 sgemm_nn_128x128_vec
  0.0            0            0          15           0          0 sgemm_nt_128x128_vec
  0.0            0            0          15           0          0 sgemm_tn_128x128_vec
  0.0            0            0          15           0          0 spool_bprop_max_overlap
  0.0            0            0          15           0          0 spool_fprop_max
  0.0            0            0          15           0          0 update_delta_3x3_4x4
  0.0            0            0          15           0          0 update_image_3x3_4x4
  0.0            0            0           1           0          0 write_size
  0.0            0            0          30           0          0 xprop_image_4x4_3x3
---------------------------------------------------------------------------------------

USER EVENTS Profile :NODE 0, CONTEXT 0, THREAD 1
```

Profile Example with Neon and Alexnet (2)



Hardware used in Experiments

- Intel Ivy Bridge and Haswell processors
- Three GPUs from 3 NVIDIA architectures

NVIDIA GPU	K80	M40	P100
CUDA capability	3.5	5.2	6.2
Global memory (MB)	11520	12288	16276
Multiprocessors (MP)	13	24	56
CUDA cores per MP	192	128	64
L2 cache (MB)	1.572	3.146	4.193
Architecture family	Kepler	Maxwell	Pascal

- All DNN libraries built with:
  - CUDA v8.0.44
  - cuDNN 5.1

Convnet Benchmarks

- Convnet tests convolutional neural network layer
- Four notable implementations include
  - AlexNet (http://vision.stanford.edu/teaching/cs231b_spring1415/slides/alexnet_tugce_kyunghee.pdf)
  - GoogLeNet (http://deeplearning.net/tag/googlenet/)
  - OverFeat (http://cilvr.nyu.edu/doku.php?id=code:start)
  - Vgg (http://www.robots.ox.ac.uk/~vgg/research/very_deep/)

- Examine similar implementations of above benchmarks in Neon and Torch, comparing three GPU architectures

Code Features

- Neon and Torch code features
  - Lines = lines of code
  - TAU routines = # routines intercepted by TAU (thread 1)
    - depends on which libraries get called

Application	Library	# lines	TAU routines
AlexNet	neon	70	26
	torch	100	66
GoogleNet	neon	92	40
	torch	100	100
Overfeat	neon	63	30
	torch	100	69
Vgg	neon	67	33
	torch	100	98

Total Execution Time

App	Library	K80	M40	P100
AlexNet	Neon	16911868.50	18358841.25	4305760.75
GoogLeNet		7279472.25	13774497.00	5520680.00
OverFeat		7864390.75	9176604.75	8059057.00
VGG		7183726.25	12744961.00	9443090.00
AlexNet	Torch	8029959.75	5078672.75	4427608.25
GoogLeNet		54639307.50	13755790.25	15526875.75
OverFeat		11296946.75	5229369.25	4976697.50
VGG		14956690.75	5059248.25	5177141.25

Instruction Throughput per # Cycles (Neon)

TAU: ParaProf: Std. Dev. Statistics - /wopr/users/roblim1/repos/neon/examples/convnet-benchmarks

File Options Windows Help

Name	Exclusive TAU...	Inclusive TAU...	Exclusive CUD...	Inclusive CUD...	Exclusive CUD...	Inclusive CUD...	Calls	Chil...
sconv_direct_updat_64x32	0.132	0.132	24,082,456.533	24,082,456.533	5,523,259.367	5,523,259.367	15	0
sconv_direct_fprop_64x128	0.116	0.116	20,765,641.067	20,765,641.067	4,856,815.517	4,856,815.517	15	0
prepare	0.002	0.002	13,276,335	13,276,335	38,476,541.5	38,476,541.5	0.5	0
sconv_winograd_3x3_4x4_32x32_D	0.055	0.055	7,719,400.533	7,719,400.533	2,182,991.333	2,182,991.333	15	0
sconv_winograd_4x4_3x3_32x32_X_Q13_N128	0.155	0.155	6,274,202.011	6,274,202.011	1,838,060.45	1,838,060.45	45	0
sconv_winograd_3x3_4x4_32x32	0.021	0.021	5,917,510.4	5,917,510.4	1,754,440.2	1,754,440.2	7.5	0
spool_bprop_max_overlap	0.032	0.032	3,546,408.356	3,546,408.356	915,999.5	915,999.5	22.5	0
sgemm_nt_128x128_vec	0.029	0.029	3,248,538.533	3,248,538.533	807,336.9	807,336.9	22.5	0
sgemm_nn_128x128_vec	0.027	0.027	2,916,601.111	2,916,601.111	695,415.756	695,415.756	22.5	0
sgemm_tn_128x128_vec	0.027	0.027	2,866,179.778	2,866,179.778	689,678.967	689,678.967	22.5	0
spool_fprop_max	0.012	0.012	1,148,160.533	1,148,160.533	325,694.211	325,694.211	22.5	0
copy_transpose	0.008	0.008	983,450.533	983,450.533	591,633.533	591,633.533	7.5	0
kernel_div_mul_mul_add_mul_add	0.033	0.033	501,396.367	501,396.367	321,351.796	321,351.796	60	0
update_image_3x3_4x4	0.017	0.017	460,891.956	460,891.956	438,536.789	438,536.789	22.5	0
xprop_image_4x4_3x3	0.032	0.032	414,720.233	414,720.233	418,346.411	418,346.411	45	0
update_delta_3x3_4x4	0.017	0.017	313,481.467	313,481.467	457,963.389	457,963.389	22.5	0
kernel_gt_lt_mul_add_mul	0.034	0.034	174,116.886	174,116.886	230,086.414	230,086.414	52.5	0
kernel_maximum_minimum_mul_add	0.022	0.022	173,788.881	173,788.881	251,537.914	251,537.914	52.5	0
bprop_filter_4x4_3x3	0.002	0.002	25,490	25,490	55,658.056	55,658.056	22.5	0
fprop_filter_4x4_3x3	0.002	0.002	18,395.256	18,395.256	52,126.311	52,126.311	22.5	0
filter_dimshuffle	0	0	13,200.933	13,200.933	8,705.667	8,705.667	7.5	0
kernel_max_sub_exp_sum_div_sub_exp_mul	0	0	10,398.2	10,398.2	27,192.2	27,192.2	7.5	0
kernel_neg_mul_safelog_mul_sum	0	0	6,558.2	6,558.2	11,792.933	11,792.933	7.5	0
kernel_sub_mul	0	0	3,350.233	3,350.233	2,542.2	2,542.2	7.5	0
kernel_onehot	0	0	2,733.567	2,733.567	2,524.4	2,524.4	7.5	0
kernel_mul	0	0	2,333.567	2,333.567	2,100.133	2,100.133	7.5	0
sconv_direct_bprop_64x128	0.085	0.085	2,080.333	2,080.333	2,385.9	2,385.9	7.5	0
write_size	0	0	11.5	11.5	1,273	1,273	0.5	0
kernel_sum_mul	0	0	3.433	3.433	75.6	75.6	7.5	0
.TAU application	11.827	2.594	0	0	0	0	0	588
Memory copy Device to Host	0	0	0	0	0	0	8	0
Memory copy Host to Device	0.093	0.093	0	0	0	0	7	0
cuCtxCreate_v2	0.446	0.446	0	0	0	0	0.5	0
cuCtxDetach	0.145	0.145	0	0	0	0	0.5	3.5
cuCtxGetApiVersion	0	0	0	0	0	0	1	0
cuCtxGetCurrent	0	0	0	0	0	0	2.5	0
cuCtxGetDevice	0.002	0.002	0	0	0	0	1,179.5	0
cuCtxGetLimit	0	0	0	0	0	0	0.5	0
cuCtxPopCurrent_v2	0	0	0	0	0	0	0.5	0
cuCtxPushCurrent_v2	0	0	0	0	0	0	0.5	0
cuCtxSetLimit	0.002	0.002	0	0	0	0	1	0

Results – Neon and Torch on Convnet

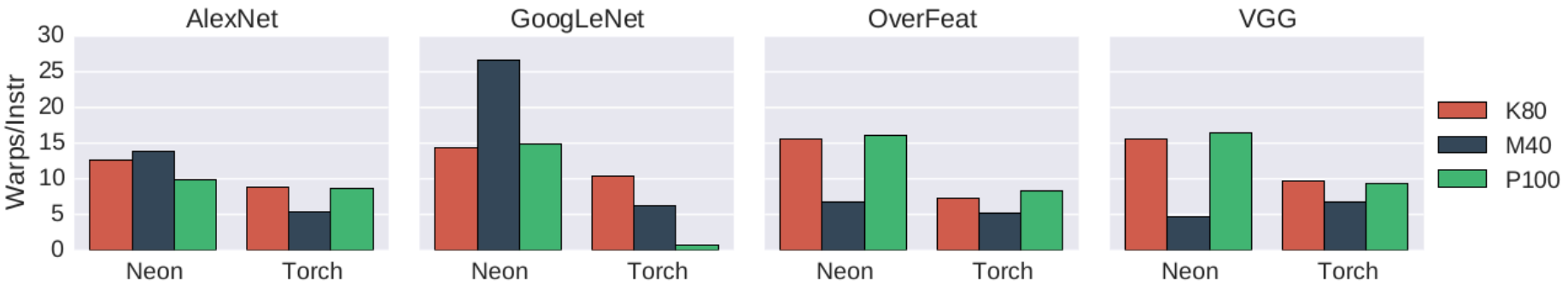


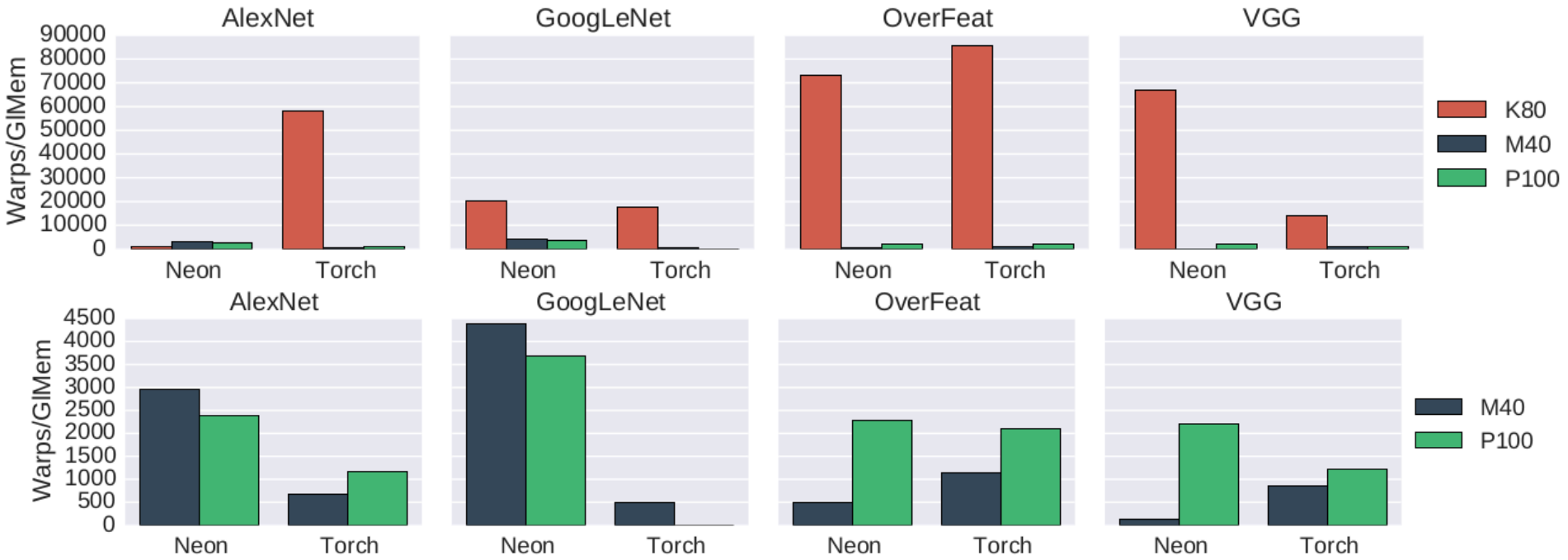
Figure: Warps executed per issued instructions, comparing Neon and Torch convnet benchmarks.

- # warps executed per issued instructions
  - Lower is better (ideally should be near zero)
  - P100 does well for AlexNet and GoogleNet
  - M40 does (very) well for OverFeat and VGG

Results – Neon and Torch on Convnet (2)

- # issued instructions per global memory operation
  - Higher is better (eliminate GM reads as much as possible)
  - K80 is clearly the outlier (bottom figure removes K80)
  - P100 does well for Neon and Torch (16 GB versus 12 GB)

Results – Neon and Torch on Convnet (3)



- # warps issued per global memory instructions
  - More is better (same reason in minimizing GM reads)
  - K80 is clearly the outlier (bottom figure removes K80)
  - P100 does well for OverFeat and VGG (128 warps/MP)
  - M40 does better in AlexNet and GoogLeNet (64 warps/MP)

Results – Neon and Torch on Convnet (4)

- Calculate metrics for individual routines
- Allows comparison of performance factors

Neon

app	metric	kernel	k80	m40	p100
alexnet	warps/ins	kernel_1h	12.89	3.73	7.32
		sconv_fp	12.89	3.73	7.32
	gld/ins	kernel_1h	0.01	0.00	0.00
		sconv_fp	0.01	0.00	0.00
googlenet	warps/ins	kernel_1h	16.25	2.34	23.77
		sconv_fp	16.25	2.34	23.77
	gld/ins	kernel_1h	0	0.00	0.02
		sconv_fp	0	0.00	0.02

Torch

app	metric	kernel	k80	m40	p100
alexnet	warps/ins	add_tensor	8.36	4.76	8.31
		cudnn.sgemm	8.34	4.25	8.06
	gld/ins	add_tensor	0.00	0.00	0.01
		cudnn.sgemm	0	0.01	0.01
googlenet	warps/ins	max_scudnn	30.20	1.42	1.25
		cudnn_dgrad	7.40	9.43	0.74
	gld/ins	max_scudnn	0	0.00	0.90
		cudnn_dgrad	0.0	0	0.80

VGG Tuning Candidates (Hot off the press!)

Arch	(warps/inst)*exec_t	(gmem/inst)*exec_t	Kernel
M40	4925585.00	43719.7865	sconv_winograd_2x2_3x3_32x32_FX_K128_W112_Q112_N64
	2165095.25	19217.5147	sconv_winograd_2x2_3x3_32x32_K128_W112_Q112_N64
	771448.48	6847.4228	kernel_maximum_minimum_mul_add
	380513.23	3377.4581	spool_fprop_max
	315010.24	11.6830	prepare
	192634.97	1709.8395	sconv_direct_fprop_64x64
P100	6208190.38	6763.7450	sconv_winograd_4x4_3x3_32x32_X_Q56_N64
	5991271.01	6527.4140	sconv_winograd_4x4_3x3_32x32_X_Q28_N64
	2828454.35	3081.5652	sconv_winograd_4x4_3x3_32x32_X_Q14_N64
	2276862.59	2480.6130	xprop_image_4x4_3x3
	2090090.03	2277.1266	sconv_winograd_2x2_3x3_32x32_FX_K128_W112_Q112_N64

Table: Candidates for performance tuning in VGG application, comparing Neon and Torch implementations.

- VGG Neon had a lot of global memory reads (previous two slides)
- Identify which routines to focus performance tuning efforts
  - `sconv_winograd` shows up a few times for both architectures!

Conclusion

- Demonstrated profiling capabilities of DNN applications using TAU Performance System
  - Real story is more complicated than this
- DL frameworks can do multi-node parallel processing
  - Do not look like your standard HPC application
  - TAU is able to support performance measurement
    - needs to support DL distributed execution models
    - Develop DL-specific performance analysis techniques
- Iterested in HPC-class applications with deep learning
  - Exascale Deep Learning and Simulation Enabled Precision Medicine for Cancer (CANDLE)
  - Livermore Big Artificial Neural Network HPC Toolkit

Acknowledgements

- J-C Vasnier of NVIDIA
  - Provide access to PSG clusters

- American Society for Engineering Education

- Franco-Américaine Fulbright Commission