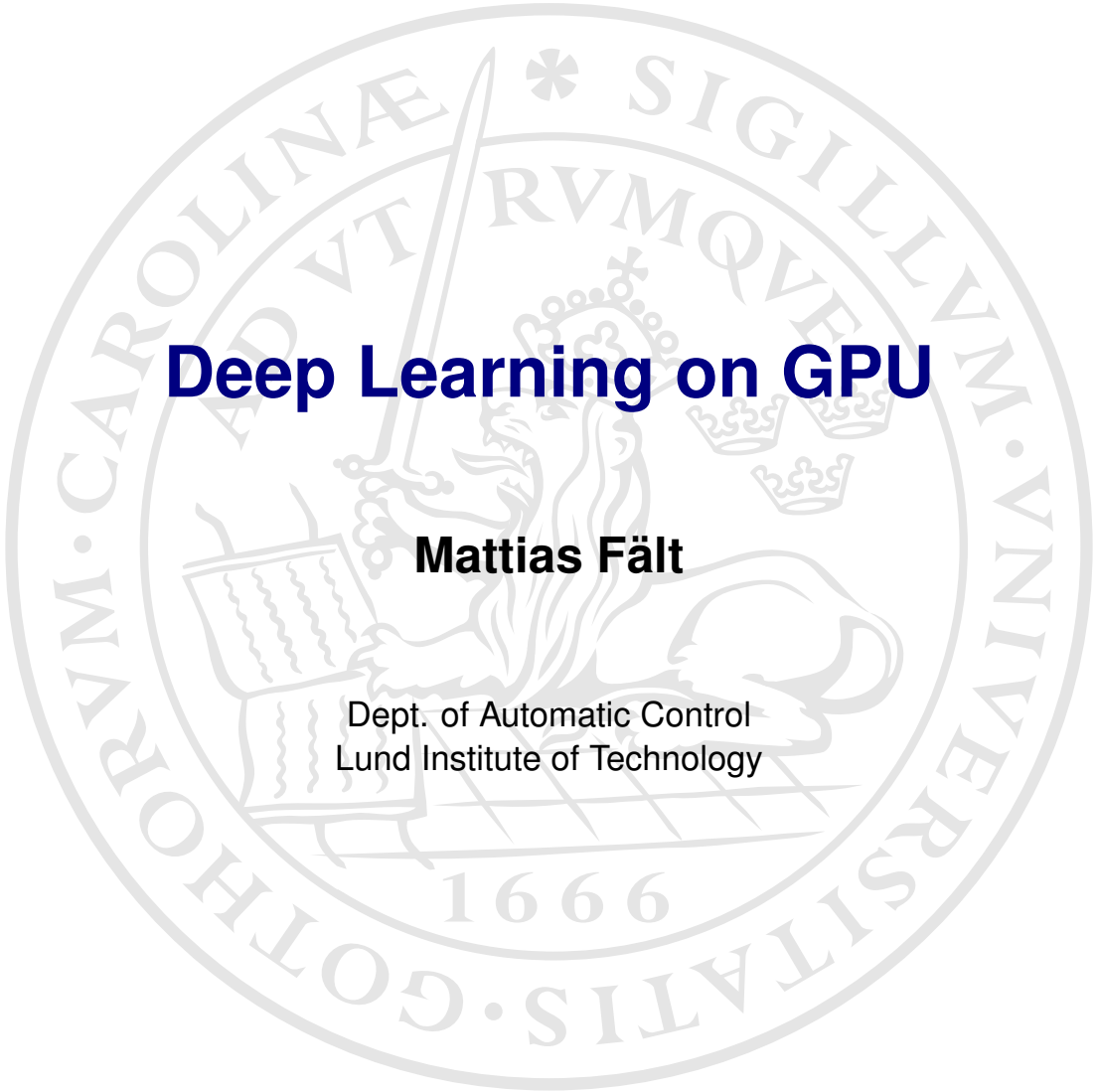


Deep Learning on GPU

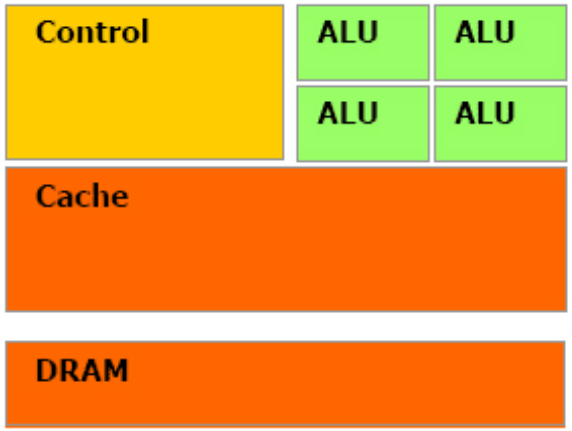
Mattias Fält

Dept. of Automatic Control
Lund Institute of Technology

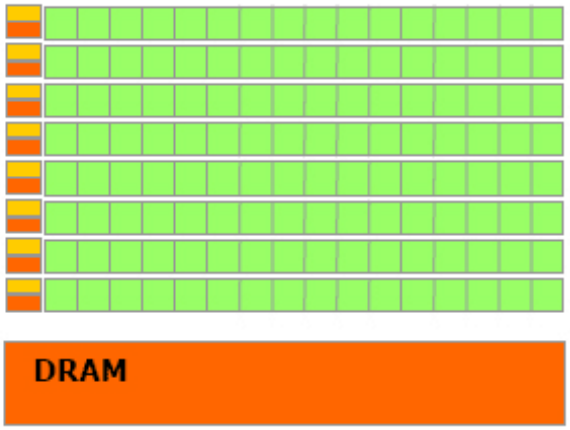
Overview

What is the difference between CPU and GPU?
What is CUDA, and how does it relate to cuBLAS and cuDNN?
How is this connected to Deep Learning and Tensorflow?
How do I run tensorflow on the GPU?
What is a TPU?

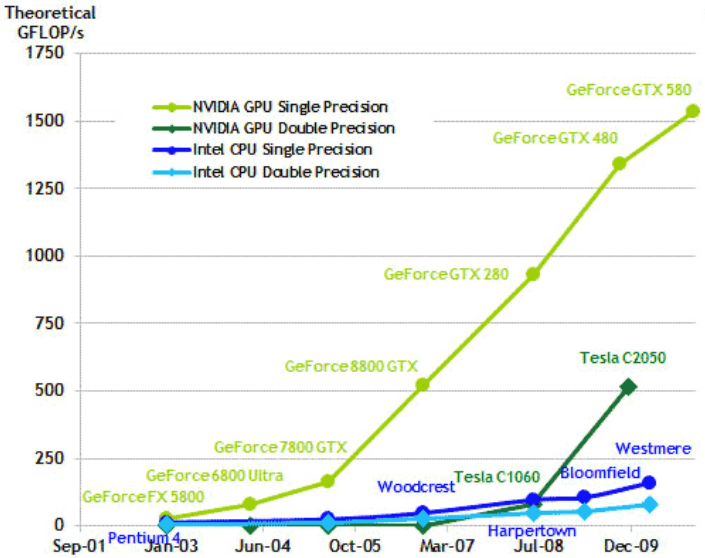
GPU vs CPU

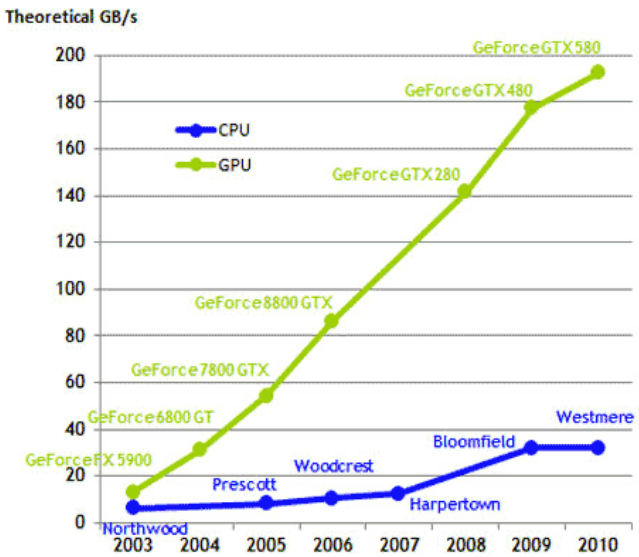


CPU



GPU





http://allegroviva.com/gpu-computing/difference-between-gpu-and-cpu

CPU (Central Processing Unit)

CPU (Intel i7 6700K, 3500kr):

- Few cores (4)
- Few threads per core (2)
- Possible to have one thread per process
- Fast clock (4GHz)
- Advanced instructions and memory management
  - Virtual Memory, Security (Protected memory from other processes)
  - Encryption (AES), Trigonometric Functions, Interruptions
  - Branch prediction, Reordering operations
  - SIMD (Single Instruction Multiple Data) (8 SP float operations per core)
- Lower Memory Bandwidth (34.2 GB/s)
- Low power (95W)
- (64 GFLOPS SP or 128 GFLOPS DP + SIMD?)

GPU (Graphics Processing Units)

GPU (Nvidia Titan X, 14000kr) (i7 in blue):

- Lots of cores (3072) (32)
  - 24 Multiprocessors
  - 128 CUDA Cores/MP
- Lots of possible threads (2048 per MP) 2(per core)
- Not one process per thread (one per MP?)
- Slower clock (≈1GHz) (4GHz)
- Simpler instructions per core
  - “Special Function Unit” to handle advanced ops (Trigonometric)
- High Memory Bandwidth (480GB/s graphics memory)
- Slower bandwidth from RAM (6.0 GB/sec) (34GB/s)
- High power (250W) (95W)
- Color LEDs! (16.8M colors) (0!)
- Up to (6600 GFLOPS SP and 206 GFLOPS DP)
- (i7: (64 GFLOPS SP or 128 GFLOPS DP or SIMD))

Conclusion:

CPU for fast and advanced dependent operations

GPU for massive parallel SIMD with shared data

CUDA

- CUDA is an API for GPGPU (General-Purpose computing on GPU)
- Released in 2007 by Nvidia

Previously, computations on GPU only possible throgh Graphics API

Alternatives exist

- Open Computing Language (OpenCL)
- Joint project by
  - Altera, AMD, **Apple**, ARM Holdings, Creative Technology, IBM, Imagination Technologies, Intel, Nvidia, Qualcomm, Samsung, Vivante, Xilinx, and ZiiLABS

CUDA Example

Create Function:

```
void saxpy(int n, float a, float *x, float *y)
{
  int i = blockIdx.x*blockDim.x + threadIdx.x;
  if (i < n) y[i] = a*x[i] + y[i];
}
```

Allocate space on GPU:

```
cudaMalloc(&d_x, N*sizeof(float));
cudaMalloc(&d_y, N*sizeof(float));
```

Copy to GPU, run and get data:

```
cudaMemcpy(d_x, x, N*sizeof(float), cudaMemcpyHostToDevice);
cudaMemcpy(d_y, y, N*sizeof(float), cudaMemcpyHostToDevice);

saxpy<<<(N+255)/256, 256>>>(N, 2.0f, d_x, d_y);

cudaMemcpy(y, d_y, N*sizeof(float), cudaMemcpyDeviceToHost);
```

cuBLAS

BLAS (Basic Linear Algebra Subroutines) library build on CUDA.
Implements several level 1-3 BLAS routines.
Names on form `cublas`T`routine`, $T \in \{S, D, C, Z\}$
Examples:

Level 1 (Scalar and Vector operations):

`cublasDaxpy`	$y = \alpha x + y$	(Alpha X Plus Y)
`cublasDdot`	$z = x \cdot y$	(Dot product)

Level 2 (Matrix - Vector operations):

`cublasDgbmv`	$y = \alpha A^{(T)} x + \beta y$	(General Banded Matrix Vector)
`cublasDspr2`	$A = \alpha(xy^T + yx^T) + A$	(Symmetric Packed Rank 2)

Level 3 (Matrix - Matrix operations):

`cublasDgemm`	$C = \alpha A^{(T)} B^{(T)} + \beta C$	(GEneral Matrix Matrix)
`cublasDtrsm`	$A^{(T)} X = \alpha B$	(TRiangular System Multiple right)

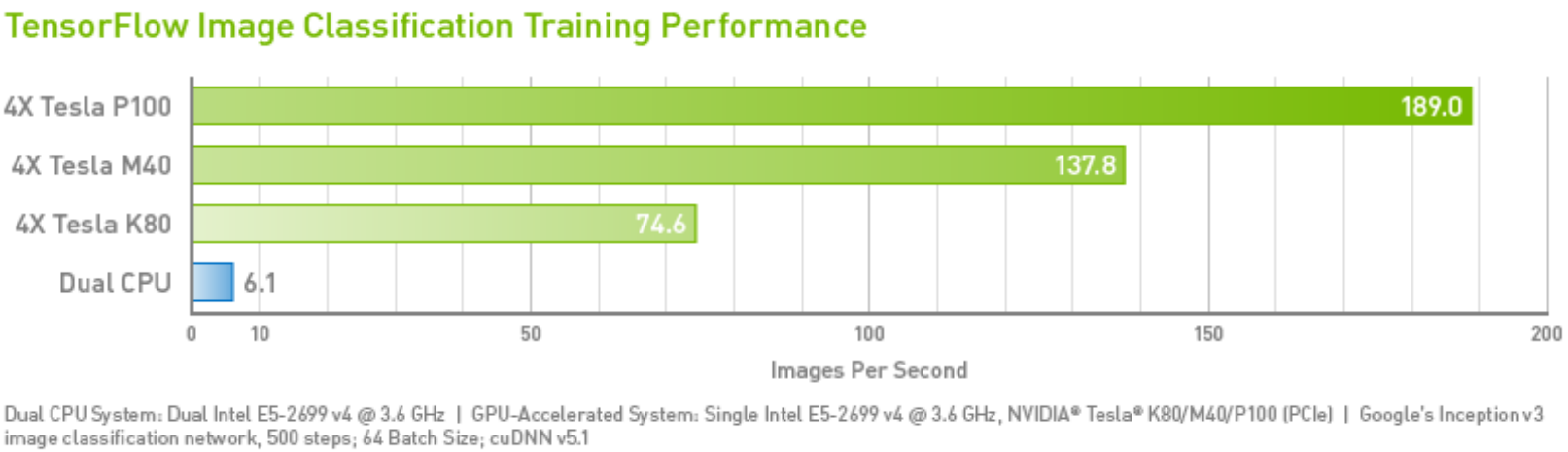
NVIDIA CUDA Deep Neural Network library (cuDNN)

A low level API and flexible, efficient, well-optimized parallel implementations of common deep learning routines for GPU.

- Efficient Primitives for Deep Learning
- Similar philosophy as BLAS.
- Can be used with e.g. Caffe, TensorFlow, Theano, Torch, CNTK
- Implements several commonly used routines:
  - Convolutions
  - Sigmoid
  - ReLU
  - Hyperbolic Tangent
  - Softmax
  - Max-pooling
  - Tensor Transformations.

TensorFlow

- TensorFlow binaries require Cuda Toolkit 8.0 and cuDNN v5 for GPU support
  - More options available if you compile TensorFlow yourself
  - Instructions: `https://www.tensorflow.org/versions/r0.11/get_started/os_setup.html`
- TensorFlow will automatically decide which operations should run on the CPU and GPU(s).
- Normal speedup is in the range 10-20x.



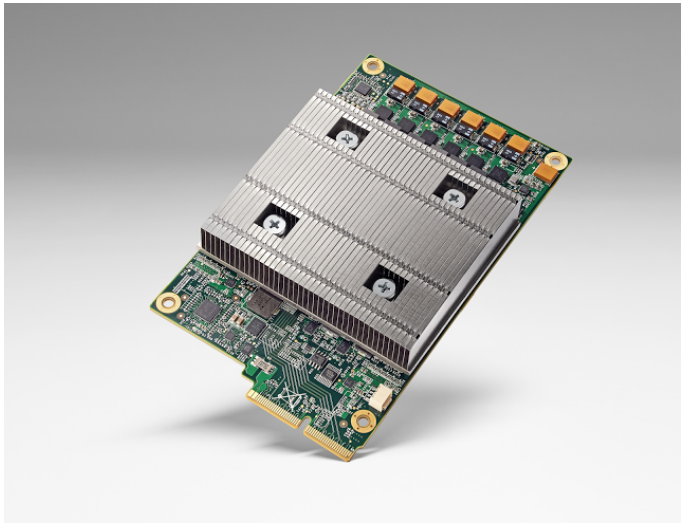
`http://www.nvidia.com/object/gpu-accelerated-applications-tensorflow-benchmarks.html`

TensorFlow

```
# Creates a graph.
a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[2, 3], name='a')
b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], shape=[3, 2], name='b')
c = tf.matmul(a, b)
# Creates a session with log_device_placement set to True.
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))
# Runs the op.
print sess.run(c)

# Output:
# Device mapping:
# /job:localhost/replica:0/task:0/gpu:0 -> device: 0,
#    name: Tesla K40c, pci bus, id: 0000:05:00.0
# b: /job:localhost/replica:0/task:0/gpu:0
# a: /job:localhost/replica:0/task:0/gpu:0
# MatMul: /job:localhost/replica:0/task:0/gpu:0
# [[ 22.  28.]
#  [ 49.  64.]]
```

TPU (Tensor Processing Unit)



- “order of magnitude better performance per watt than standard solutions”
- All Google TPUs “can find all the text in the Street View database in less than five days”
- One can process more than 100 million Google photos per day.
- Very secret:
  - Used for training or evaluation?
  - “reduced computational precision”: 32bit, 16bit, 8bit?

https://cloudplatform.googleblog.com/2016/05/Google-supercharges-machine-learning-tasks-with-custom-chip.html

Homework

Either

- Run one of your own examples on a CPU and GPU and compare the times. Do all computations run on the GPU?
- `https://www.tensorflow.org/versions/r0.11/how_tos/using_gpu/index.html`

Or

- Read: TensorFlow: A system for large-scale machine learning
  - `https://arxiv.org/abs/1605.08695`