Deep Learning on AWS

No Math, I promise!

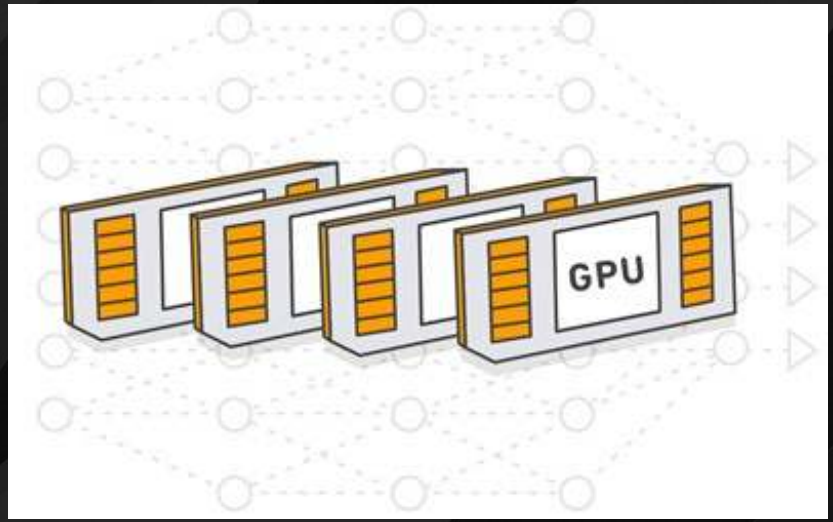
Julien Simon
Principal Technical Evangelist
Amazon Web Services

julsimon@amazon.fr
@julsimon

22/11/2016

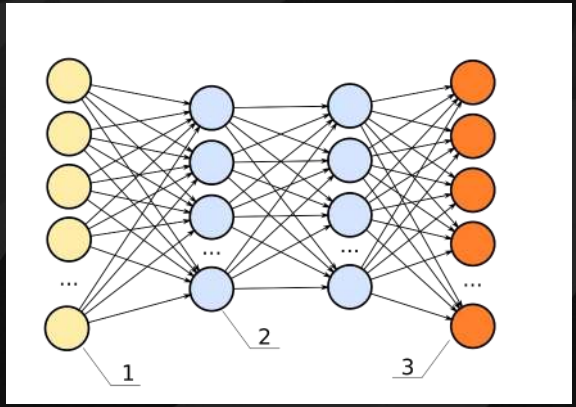
Agenda

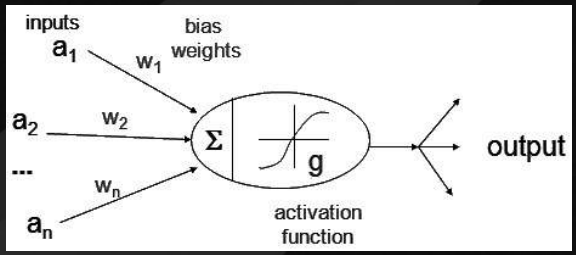
- Neural networks
- Recommendation @ Amazon.com
- GPU instances & Nvidia CUDA
- Amazon DSSTNE
- Deep Learning AMI

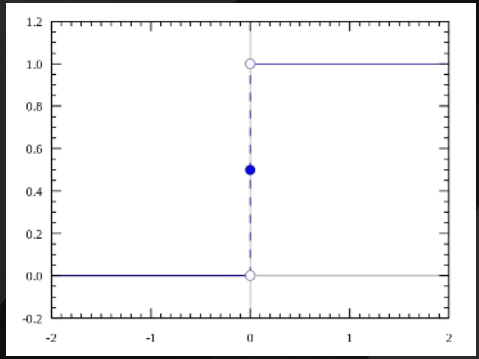


Neural networks

- Nodes, weights and layers
- Activation function
- Forward propagation
- Training
  - Input layer: 1 to n data samples
  - Output layer: the expected result
  - Use backpropagation to minimize error by adjusting weights
- Predicting
  - Input layer: 1 data sample
  - Output layer: the predicted result



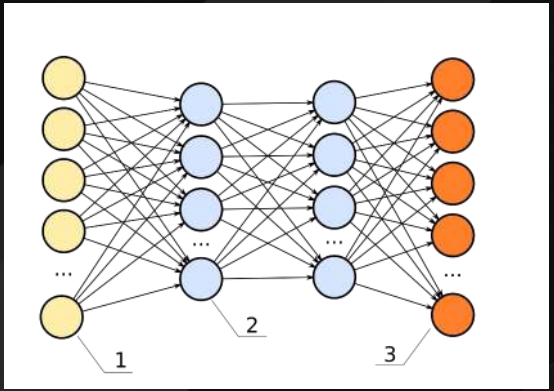




http://machinelearningmastery.com/what-is-deep-learning/
https://devblogs.nvidia.com/parallelforall/deep-learning-nutshell-core-concepts/

Neural networks for product recommendation

- The input and output layers correspond to products (“*if you bought this, you’ll like this*”)
- Amazon.com has hundreds of millions of products → Recommendation problems require very large input & output layers
- Training and recommendation are performed using matrix operations, which can best be scaled using GPUs.



https://commons.wikimedia.org/

- Most of the matrix is zero-filled, because only a fraction of products are recommended in the dataset

Recommendation @ Amazon.com

Generating Recommendations at Amazon Scale with Apache Spark and Amazon DSSTNE

by Kiuk Chung | on 09 JUL 2016 | Permalink | Comments

Kiuk Chung is a Software Development Engineer with the Amazon Personalization team

In Personalization at Amazon, we use neural networks to generate personalized product recommendations for our customers. Amazon's product catalog is huge compared to the number of products that a customer has purchased, making our datasets extremely sparse. And with hundreds of millions of customers and products, our neural network models often have to be distributed across multiple GPUs to meet space and time constraints.

For this reason, we have created and open-sourced DSSTNE, the Deep Scalable Sparse Tensor Neural Engine, which runs entirely on the GPU. We use DSSTNE to train neural networks and generate recommendations that power various personalized experiences on the retail website and Amazon devices.

https://aws.amazon.com/blogs/big-data/generating-recommendations-at-amazon-scale-with-apache-spark-and-amazon-dsstne/

Amazon DSSTNE (aka ‘Destiny’)

- Deep Scalable Sparse Tensor Network Engine
- Open source library for deep neural networks using GPUs https://github.com/amznlabs/amazon-dsstne
- Multi-GPU scale for training and prediction
- Larger networks than are possible with a single GPU
- Optimized for fast performance on sparse datasets
- Human-readable networks (JSON file)
- Can run locally or on AWS

AWS GPU Instances

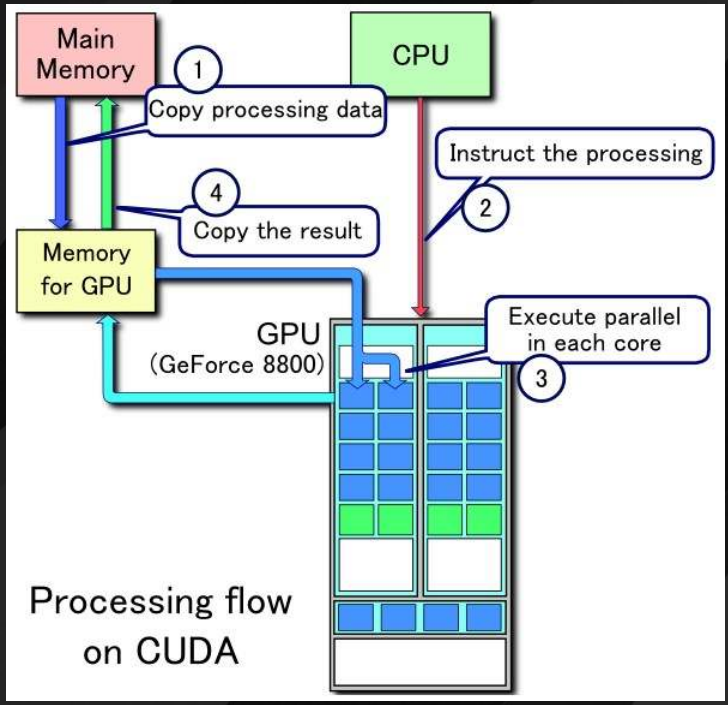
- g2 (2xlarge, 8xlarge)
  - 32 vCPUs, 60 GB RAM
  - 4 NVIDIA K520 GPUs
  - 16 GB of GPU memory, 6144 CUDA cores
- p2 (xlarge, 8xlarge, 16xlarge)
  - Launched in 09/16
  - 64 vCPUs, 732 GB RAM
  - 16 NVIDIA GK210 GPUs
  - 192 GB of GPU memory, 39936 CUDA cores
  - 20 Gbit/s networking

EC2 Instance Type	Total
g2.2xlarge	$0.65/hr
g2.8xlarge	$2.60/hr
p2.8xlarge	$7.20/hr
p2.xlarge	$0.90/hr
p2.16xlarge	$14.40/hr

https://aws.amazon.com/blogs/aws/new-g2-instance-type-with-4x-more-gpu-power/
https://aws.amazon.com/blogs/aws/new-p2-instance-type-for-amazon-ec2-up-to-16-gpus/

Nvidia CUDA architecture

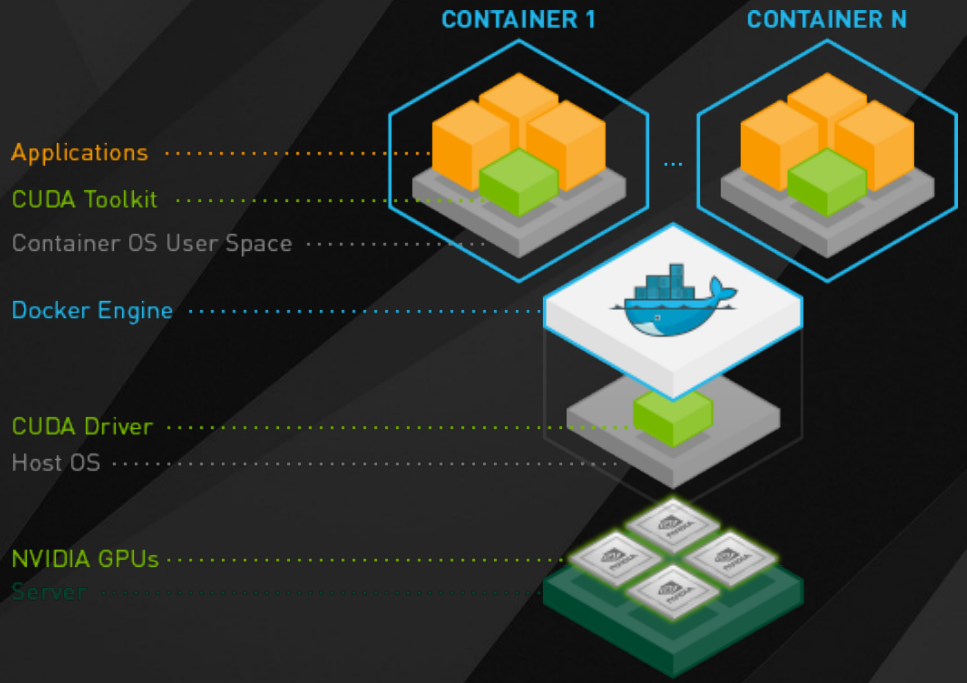
- CUDA is a parallel computing platform and application programming interface model created by Nvidia
- CUDA toolkit to build applications (compiler, etc.)
- CUDA drivers to manage GPUs



http://www.nvidia.com/object/cuda_home_new.html
https://en.wikipedia.org/wiki/CUDA

Deploying CUDA apps with Docker

- Docker containers are hardware-agnostic and platform-agnostic
- Installing Nvidia drivers inside containers breaks this model
- nvidia-docker solves this issue by using host drivers



https://www.docker.com/what-docker
https://github.com/NVIDIA/nvidia-docker

Full details here: https://github.com/juliensimon/aws/blob/master/dsstne/demo-docker.txt

Building DSSTNE

DSSTNE has quite a few dependencies
(see https://github.com/amznlabs/amazon-dsstne)

1. Install them manually
2. Use the DSSTNE AMI (CUDA 7.0)
3. Launch a container installing all dependencies in its Docker file (CUDA 8.0, the latest version)
   https://github.com/tristanpenman/docker-dsstne

Demo: Amazon Destiny

http://grouplens.org/datasets/movielens/

27,000 movies

138,000 users

20 million movie recommendations

(Matrix is 99.5% sparse)

→Train a neural network

Input layer: 27,000 neurons
(1 for each movie)

1 hidden layer: 128 neurons

Output layer: 27,000 neurons
(1 for each movie)

→ Recommend 10 movies per user

	Alice	Bob	Charlie	David	Ernest
Star Wars		1		1	1
Lord of the Rings	1		1		1
Inception		1			
Bambi	1			1	
Pretty Woman	1				1

	Alice	Bob	Charlie	David	Ernest
Star Wars	0.23	1	0.12	1	1
Lord of the Rings	1	0.34	1	0.89	1
Inception	0.8	1	0.43	0.76	0.45
Bambi	1	0.42	0.5	1	0.34
Pretty Woman	1	0.09	0.67	0.04	1

Training & predicting

```
# Fetch the Movie Lens dataset and the neural network config file
wget https://s3.amazonaws.com/amazon-dsstne-data/movielens/ml20m-all
wget https://s3.amazonaws.com/amazon-dsstne-data/movielens/config.json
# Format the input and output data for training
generateNetCDF -d gl_input -i ml20m-all -o gl_input.nc -f features_input -s samples_input -c
generateNetCDF -d gl_output -i ml20m-all -o gl_output.nc -f features_output -s samples_input -c
# Train the model on 8 GPUs
mpirun -np 8 train -c config.json -i gl_input.nc -o gl_output.nc -n gl.nc -b 256 -e 10
# Predict the results
predict -b 1024 -d gl -i features_input -o features_output -k 10 -n gl.nc -f ml20m-all -s recs -r ml20m-all
```

http://www.unidata.ucar.edu/software/netcdf
https://www.open-mpi.org

Amazon Destiny vs Google TensorFlow

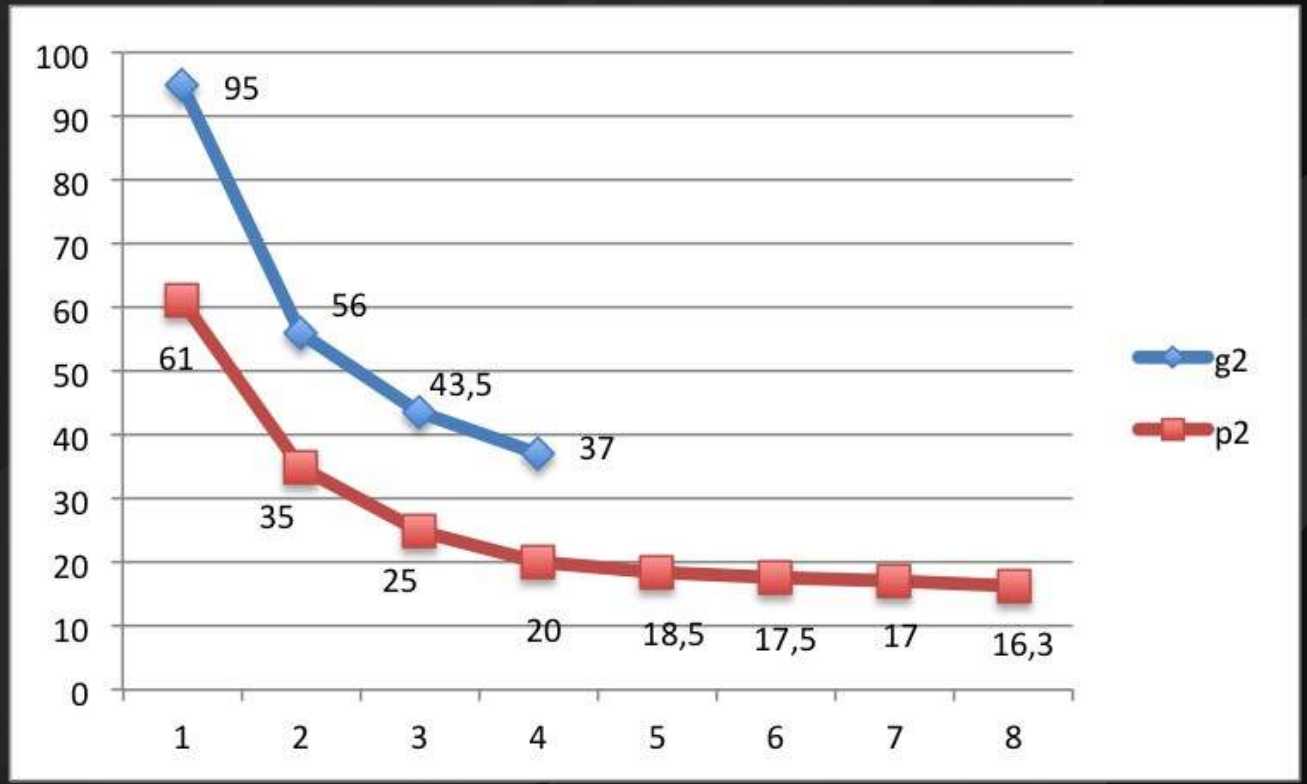
First DSSTNE Benchmarks TLDR: Up to Almost 15x Faster than TensorFlow

"DSSTNE on a single virtualized K520 GPU (released in 2012) is faster than TensorFlow on a bare metal Tesla M40 (released in 2015)"

"TensorFlow does not provide the automagic model parallelism provided by DSSTNE"

https://medium.com/@scottlegrand/first-dsstne-benchmarks-tldr-almost-15x-faster-than-tensorflow-393dbeb80c0f

Training Amazon Destiny on multiple GPUs



Movie Lens 20M, g2.8xlarge vs p2.16xlarge
mpirun –np <n> train -c config.json -i gl_input.nc -o gl_output.nc -n gl.nc -b 256 -e 10

AWS Deep Learning AMI

- Deep Learning Frameworks – 5 popular Deep Learning Frameworks (MXNet, Caffe, Tensorflow, Theano, and Torch) all prebuilt and pre-installed
- Pre-installed components – Nvidia drivers, cuDNN, Anaconda, Python2 and Python3
- AWS Integration – Packages and configurations that provide tight integration with Amazon Web Services like Amazon EFS (Elastic File System)

Resources

Big Data Architectural Patterns and Best Practices on AWS
https://www.youtube.com/watch?v=K7o5OlRLtvU

Real-World Smart Applications With Amazon Machine Learning
https://www.youtube.com/watch?v=sHJx1KJf8p0

Deep Learning: Going Beyond Machine Learning
https://www.youtube.com/watch?v=Ra6m70d3t0o

DSSTNE: A new Deep Learning Framework For Large Sparse Datasets
https://www.youtube.com/watch?v=LbYR6Mzq6FE

AWS Big Data blog: https://blogs.aws.amazon.com/bigdata/

Thank you!

Julien Simon
Principal Technical Evangelist
Amazon Web Services

julsimon@amazon.fr
@julsimon