集群部署说明

预备知识

1 ： tensorflow集群的节点有两种worker server(ws)和parameter server(ps)；其中ws有大规模计算，可以使用gpu加速，而ps使用cpu即可。

2 ： tensorflow只支持NVIDIA的显卡作为GPU;并且有如下条件：

- CUDA® Toolkit 8.0
- cuDNN v6.0
- GPU card with CUDA Compute Capability 3.0 or higher

详见：1：tensorflow只支持NVIDIA的显卡作为GPU

3：一个主机目前一个主机最多8个显卡

4：GPU并行有两种方式，有几种方式单机多显卡；多机单显卡；多机多显卡。相同的GPU数据量下单机多显卡性能要高于多机单显卡。因为高性能计算主要瓶颈在网络带宽；cpu、gpu、ssd的带宽远大于网络带宽。

参考：3：高性能计算下网络带宽是瓶颈，SSD带宽远大于网络带宽 、4：高性能计算下网络带宽是瓶颈，cpu、gpu带宽远大于网络带宽

5：TensorFlow的第一个分布式版本0.8，使用16块GPU可达单GPU的15倍提速，在50块GPU时可达到40倍提速，分布式的效率很高；但是100快GPU只相当于56倍速度。

详见：5：Tensorflow0.8 GPU并行扩展效率

注：TensorFlow版本更新很快，社区很活跃，15年底才开源；目前已经是1.5版本了。并且GPU并行扩展的效率本身在改进，未找到后续版本关于GPU并行扩展的效率的说明。

6：已经有其它企业和组织宣称可以将GPU几乎线性扩展的

a) 6：Scaling TensorFlow and Caffe to 256 GPUs

b) 7：Poseidon: An Efficient Communication Interface for Distributed Deep Learning on GPU Clusters

生产环境分布式GPU配置

GPU

只能选择Navidia显卡，在Tensorflow官网中提到的型号有Tesla K80、Titan X (Maxwell and Pascal), M40, P100; 见：optimizing_for_gpu

CPU

Tensorflow官网推荐的是Intel® Xeon® 和Intel® Xeon Phi™，因为带了Intel® MKL-DNN;专门针对深度学习做了优化：见：optimizing_for_cpu

整体配置

只找到三个资料有说明较为完整的集群配置的

1：Mellanox 公司：Reference Deployment Guide for TensorFlow with an NVIDIA GPU Card over Mellanox 100 GbE Network

1个参数服务器；4个Worker服务器；所有主机的内存为256G；ws具体用几个GPU没有说。

Parameter server:

- E5-2650V4, 12 cores @ 2.2GHz, 30M L2 cache, 9.6GT QPI
- 256GB RAM: 16 x 16 GB DDR4 dual rank
- One Mellanox ConnectX-4 VPI 100GbE adapter
- Ubuntu 16.04 x86_64 used as OS on all servers

Four Worker servers, each containing:

- E5-2650V4, 12 cores @ 2.2GHz, 30M L2 cache, 9.6GT QPI
- 256GB RAM: 16 x 16 GB DDR4 dual rank
- One or more CUDA-Capable GPU Cards with Compute Capability 3.0 or higher (http://developer.nvidia.com/cuda-gpus)
- One Mellanox ConnectX-4 VPI 100GbE adapter
- Ubuntu 16.04 x86_64 used as OS on all servers

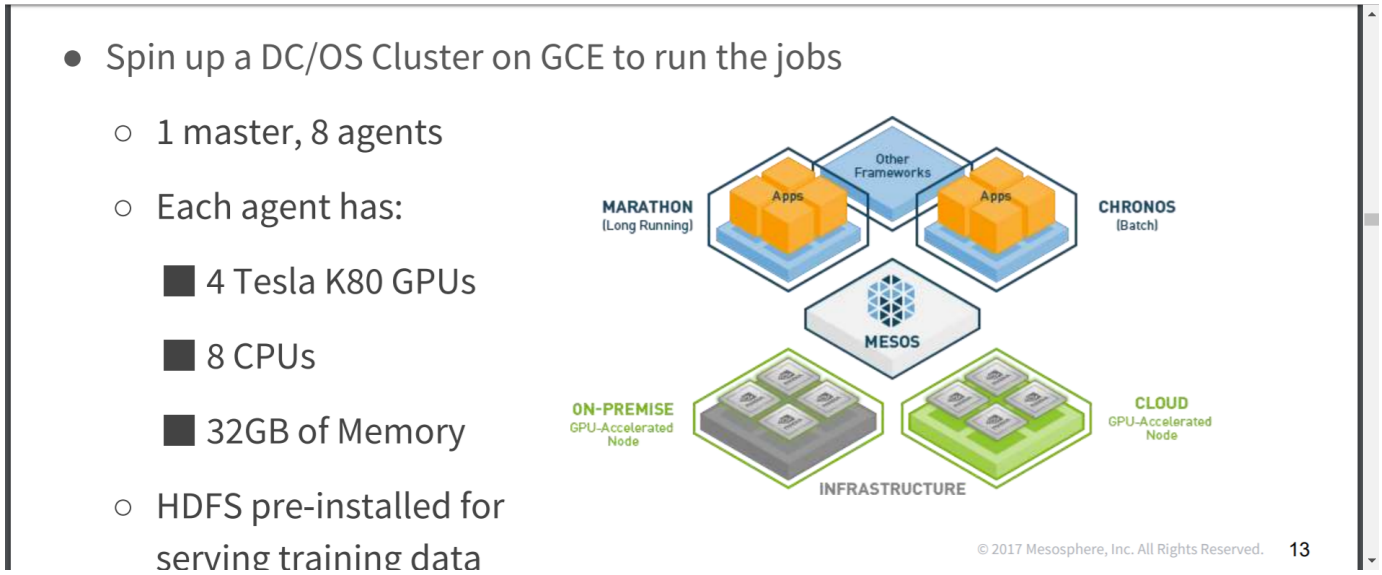
Networking

- SN2700 32 ports 100GbE QSFP28
- Mellanox LinkX MCP1600-Cxxx series cables for connect servers to 100GbE network

详见：https://community.mellanox.com/docs/DOC-2823#jive_content_id_Equipment

2：Mesosphere数据中心操作系统（DCOS）：Running Distributed TensorFlow with GPUs on Mesos with DC/OS

1个参数服务器，8个Worker服务器；每个ws有4个Tesla K80 GPUs、8个CPU、32G内存；参数服务器配置没有说明。（注：8台ws，每台4个Tesla K80 GPUs；一共是32个GPU）



详见：Running Distributed TensorFlow on DC/OS ，Running Distributed TensorFlow on DCOS.pdf

3：tensorflow官网：Details for Amazon EC2 Distributed (NVIDIA® Tesla® K80)

8个ws，每个ws 8个Tesla® K80 GPU;一共64个。p2.8xlarge实例配置为8CPU、32核，488G内存。（注：这个配置好像很夸张，不过ps是共享的ws主机，即ps和ws在相同主机上）

- **Instance type**: p2.8xlarge
- **GPU:** 8x NVIDIA® Tesla® K80
- **OS:** Ubuntu 16.04 LTS
- **CUDA / cuDNN:** 8.0 / 5.1
- **TensorFlow GitHub hash:** b1e174e
- **Benchmark GitHub hash:** 9165a70
- **Build Command:**`bazel build -c opt --copt=-march="haswell" --config=cuda //tensorflow/tools/pip_package:build_pip_package`
- **Disk:** 1.0 TB EFS (burst 100 MB/sec for 12 hours, continuous 50 MB/sec)
- **DataSet:** ImageNet
- **Test Date:** May 2017

To simplify server setup, EC2 instances (p2.8xlarge) running worker servers also ran parameter servers. Equal numbers of parameter servers and worker servers were used with the following exceptions:

- InceptionV3: 8 instances / 6 parameter servers
- ResNet-50: (batch size 32) 8 instances / 4 parameter servers
- ResNet-152: 8 instances / 4 parameter servers

详见：https://www.tensorflow.org/performance/benchmarks#details_for_amazon_ec2_distributed_nvidia_tesla_k80

亚马逊p2.8xlarge的配置如下：（来源：https://aws.amazon.com/cn/blogs/aws/new-p2-instance-type-for-amazon-ec2-up-to-16-gpus/）

Instance Name	GPU Count	vCPU Count	Memory	Parallel Processing Cores	GPU Memory	Network Performance
p2.xlarge	1	4	61 GiB	2,496	12 GiB	High
p2.8xlarge	8	32	488 GiB	19,968	96 GiB	10 Gigabit
p2.16xlarge	16	64	732 GiB	39,936	192 GiB	20 Gigabit

存在问题

1：ps和ws比例多少合适；目前看到的有（1：1；1：2；1：4；1：8）。tensorflow官网的测试的ps:ws 为1：2和3：4.这里ps共用了ws主机。

2：ps和ws的中内存多大合适；上面几个配置有32G也有256G甚至488G。

阿里云上说：需要高性能Nvidia GPU计算卡，内存不小于两倍的显存

3：如果ws使用的是GPU,那么cpu是不是配置可以差一点。

开发环境分布式GPU配置

开发环境使用单机双GPU即可测试集群；在同一个主机运行1个ps和两个2ws。

GPU：两个Tesla M40

内存：128G

CPU：32核

参考2018-1-15日刘总给的意见：

CPU：Intel E5 2个CPU 24核+

内存：DDR4 64G

GPU：2个 GTX1080TI 公版

磁盘：SSD 512G、HDD 2T

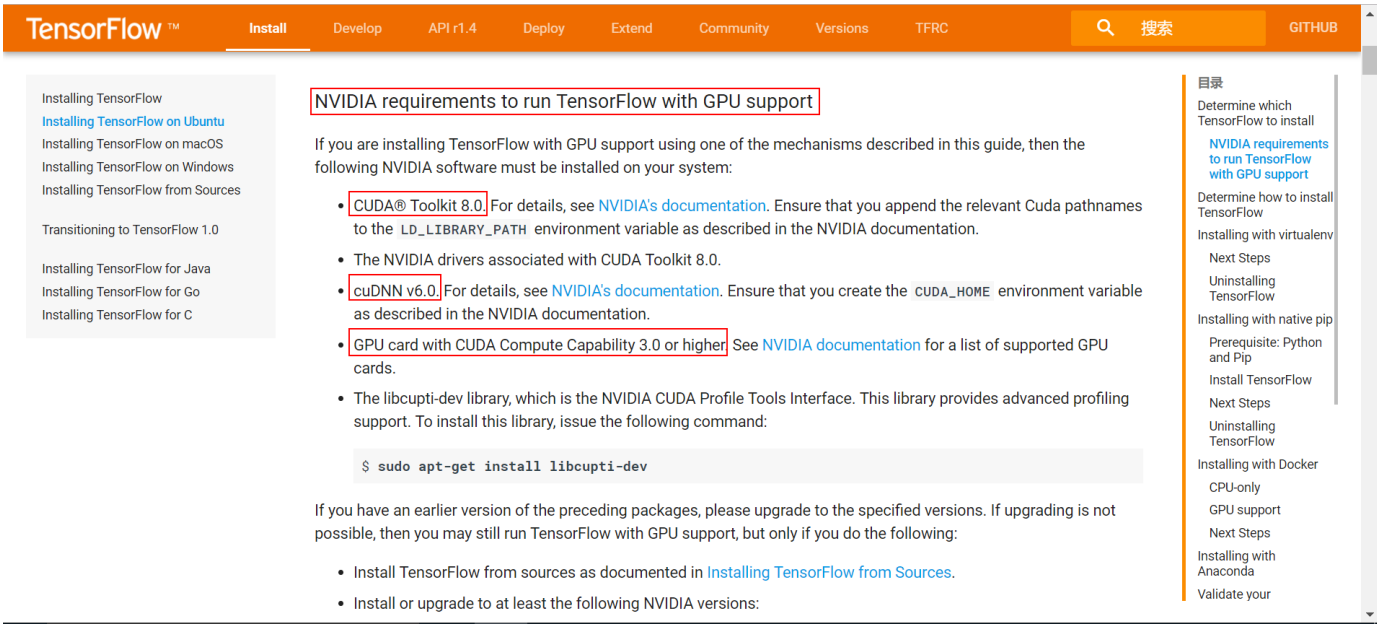
操作系统： Ubuntu 16.04 LTS

其他参考：如何配置一台适用于深度学习的工作站？

附：相关资料

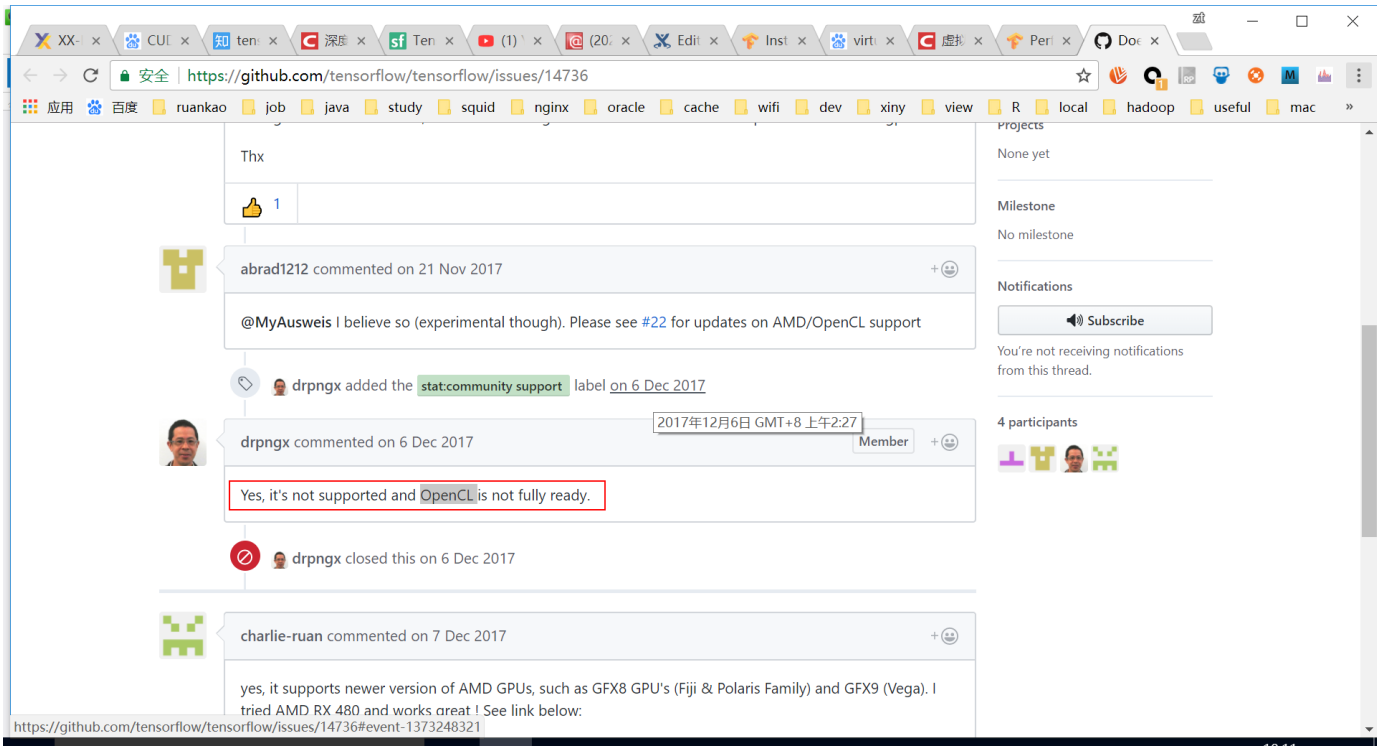
1：tensorflow只支持NVIDIA的显卡作为GPU

https://www.tensorflow.org/install/install_linux#nvidia_requirements_to_run_tensorflow_with_gpu_support



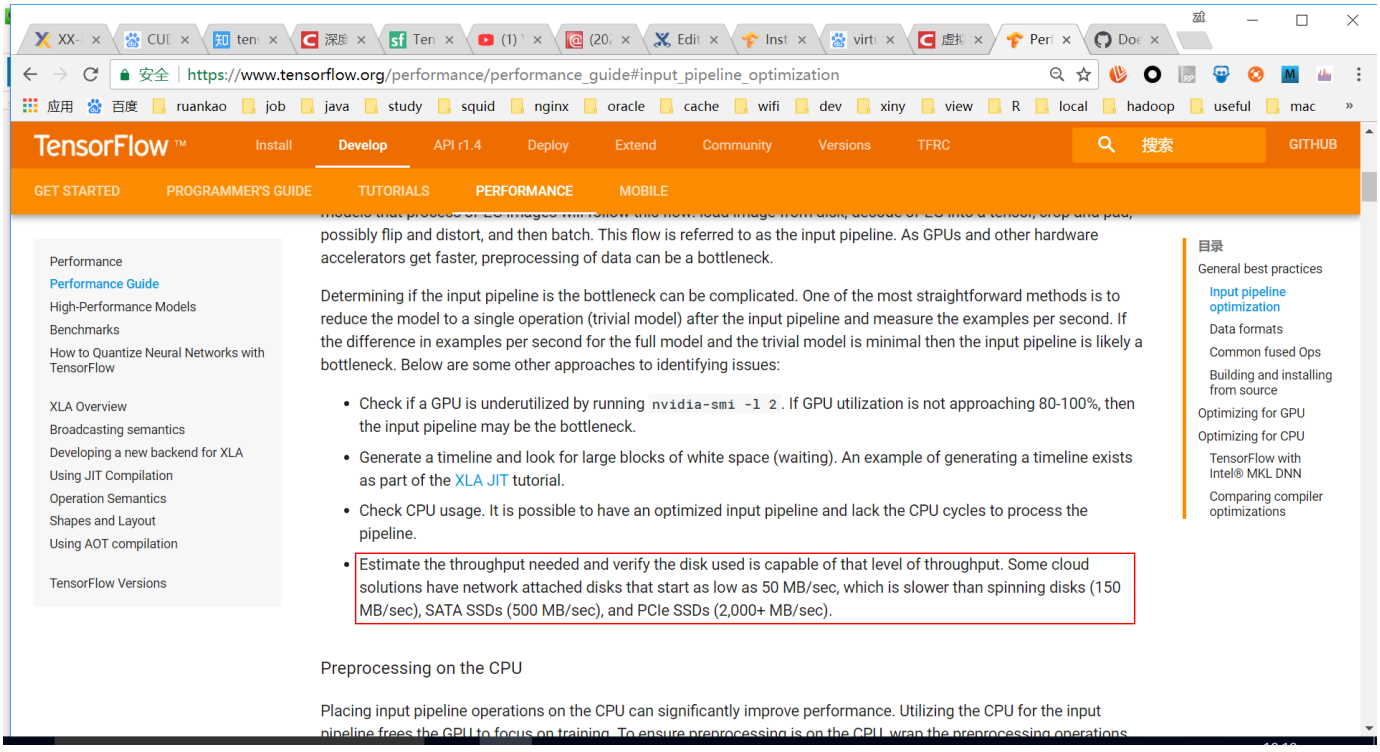
2： 有通过OpenCL支持AMD的GPU，但没有CUDA成熟；也没有完全支持

https://github.com/tensorflow/tensorflow/issues/14736



3：高性能计算下网络带宽是瓶颈，SSD带宽远大于网络带宽

https://www.tensorflow.org/performance/performance_guide#input_pipeline_optimization



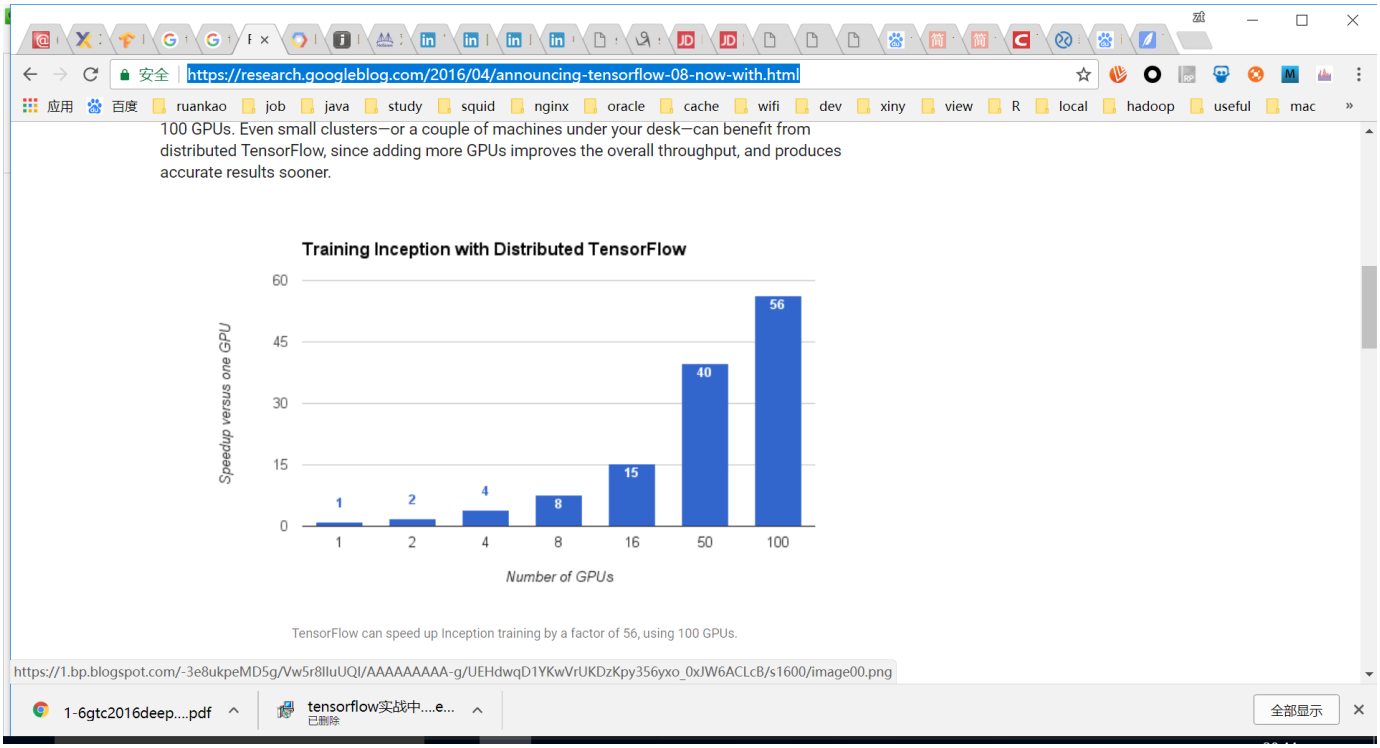
4：高性能计算下网络带宽是瓶颈，cpu、gpu带宽远大于网络带宽

https://www.jianshu.com/p/937a0ce99f56

5：Tensorflow0.8 GPU并行扩展效率

https://research.googleblog.com/2016/04/announcing-tensorflow-08-now-with.html；

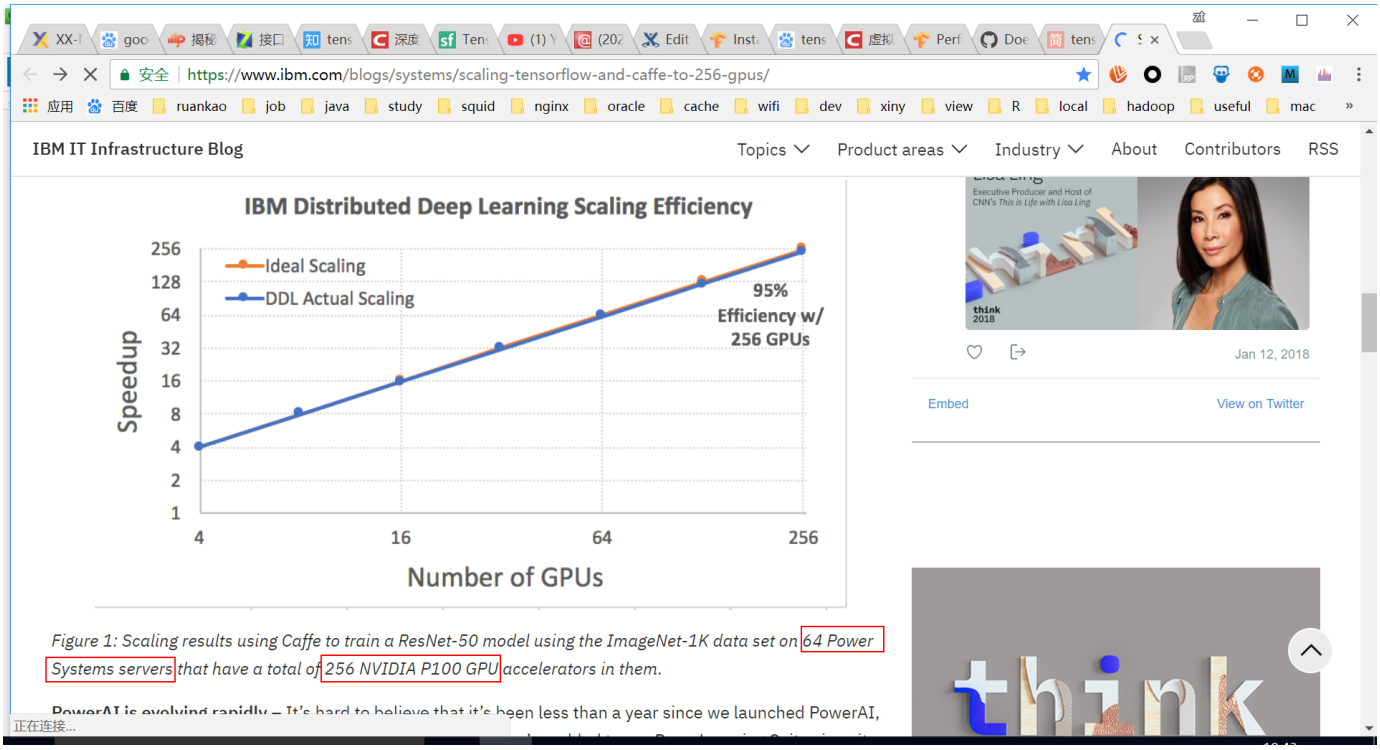
在GPU是8之前是线性的，应该是在单机多显卡；50快GPU还不错，有40倍的提升；后来性能下降就很快了，100块GPU只有56倍提升。



6: Scaling TensorFlow and Caffe to 256 GPUs

https://www.ibm.com/blogs/systems/scaling-tensorflow-and-caffe-to-256-gpus/

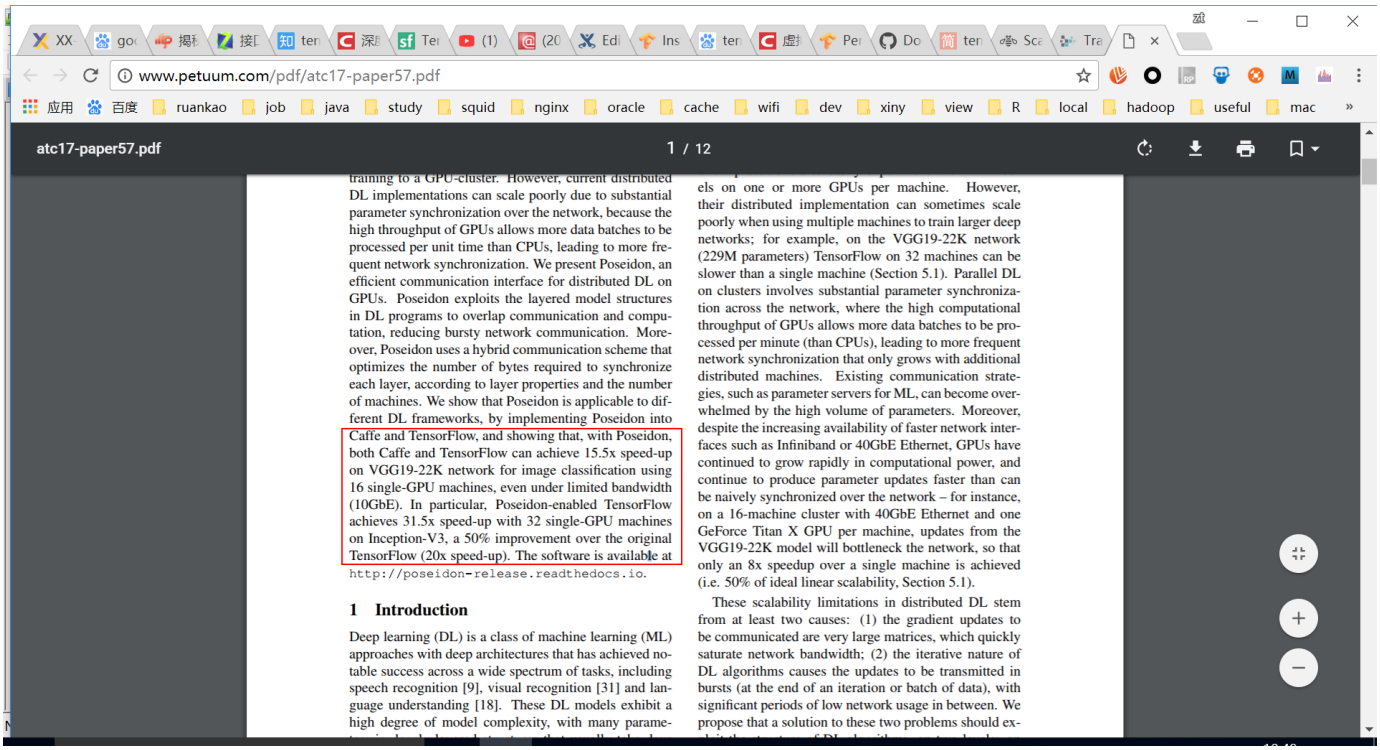
几乎是线性扩展，256块GPU，性能只损失百分之五。



7：Poseidon: An Efficient Communication Interface for Distributed Deep Learning on GPU Clusters

http://www.petuum.com/pdf/atc17-paper57.pdf

文中说通过Poseidon，32块GPU可以获得31.5倍的速度；原生的Tensorflow只有20倍的速度(注:跟tensorflow 0.8官方测试有出入，不过测试是不同的网络)



training to a GPU-cluster. However, current distributed DL implementations can scale poorly due to substantial parameter synchronization over the network, because the high throughput of GPUs allows more data batches to be processed per unit time than CPUs, leading to more frequent network synchronization. We present Poseidon, an efficient communication interface for distributed DL on GPUs. Poseidon exploits the layered model structures in DL programs to overlap communication and computation, reducing bursty network communication. Moreover, Poseidon uses a hybrid communication scheme that optimizes the number of bytes required to synchronize each layer, according to layer properties and the number of machines. We show that Poseidon is applicable to different DL frameworks, by implementing Poseidon into Caffe and TensorFlow, and showing that, with Poseidon, both Caffe and TensorFlow can achieve 15.5x speed-up on VGG19-22K network for image classification using 16 single-GPU machines, even under limited bandwidth (10GbE). In particular, Poseidon-enabled TensorFlow achieves 31.5x speed-up with 32 single-GPU machines on Inception-V3, a 50% improvement over the original TensorFlow (20x speed-up). The software is available at `http://poseidon-release.readthedocs.io`.

1 Introduction

Deep learning (DL) is a class of machine learning (ML) approaches with deep architectures that has achieved notable success across a wide spectrum of tasks, including speech recognition [9], visual recognition [31] and language understanding [18]. These DL models exhibit a high degree of model complexity, with many parame-

els on one or more GPUs per machine. However, their distributed implementation can sometimes scale poorly when using multiple machines to train larger deep networks; for example, on the VGG19-22K network (229M parameters) TensorFlow on 32 machines can be slower than a single machine (Section 5.1). Parallel DL on clusters involves substantial parameter synchronization across the network, where the high computational throughput of GPUs allows more data batches to be processed per minute (than CPUs), leading to more frequent network synchronization that only grows with additional distributed machines. Existing communication strategies, such as parameter servers for ML, can become overwhelmed by the high volume of parameters. Moreover, despite the increasing availability of faster network interfaces such as Infiniband or 40GbE Ethernet, GPUs have continued to grow rapidly in computational power, and continue to produce parameter updates faster than can be naively synchronized over the network – for instance, on a 16-machine cluster with 40GbE Ethernet and one GeForce Titan X GPU per machine, updates from the VGG19-22K model will bottleneck the network, so that only an 8x speedup over a single machine is achieved (i.e. 50% of ideal linear scalability, Section 5.1).

These scalability limitations in distributed DL stem from at least two causes: (1) the gradient updates to be communicated are very large matrices, which quickly saturate network bandwidth; (2) the iterative nature of DL algorithms causes the updates to be transmitted in bursts (at the end of an iteration or batch of data), with significant periods of low network usage in between. We propose that a solution to these two problems should ex-