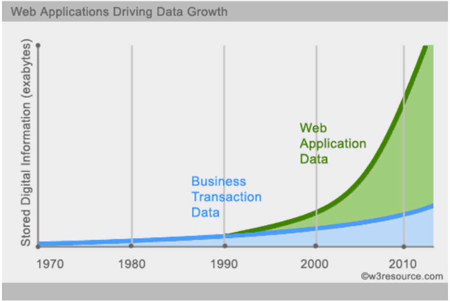
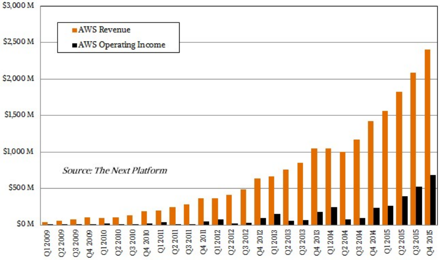
SMT Presentations: Computation

Motivation

Big data



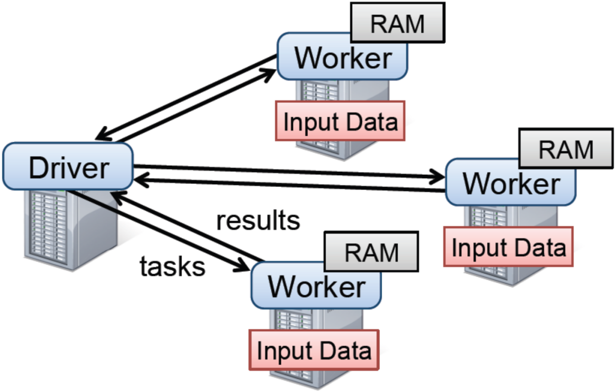
Cloud computing



Problem

Datasets no longer fit on a single machine!

Distributed Computing



New concerns:

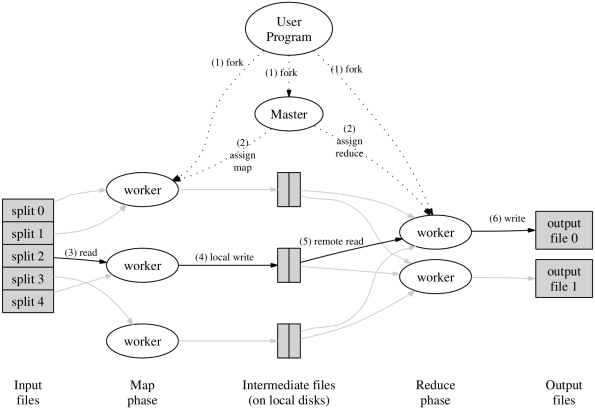
1. Consistency
2. Availability
3. Partition tolerance

CAP Theorem [GL02]

Choose two

MapReduce [DG08]

- Job-level dependency tracking
- Disk I/O every iteration



SparkNet

Training Deep Networks in Spark [MNSJ15]

Feynman Liang

CUED

May 17, 2016

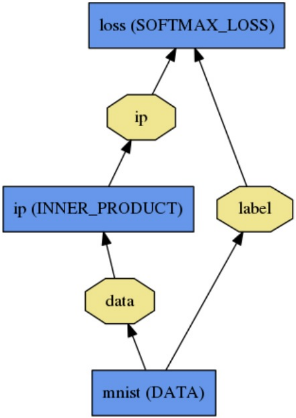
Goals

- "Our goal is **not to outperform custom computational frameworks** but rather to propose a system that can be **easily implemented in popular batch frameworks** and that performs nearly as well..."
- "...integration with a fast and general engine for big data processing such as Spark **allows researchers and practitioners to draw from a rich ecosystem of tools to develop and deploy their models**"
- "In SparkNet, **training a deep network on the output of a SQL query, or a graph computation, or a streaming data source** is straightforward"

Caffe [JSD+14]

- Separation of concerns
  - Model specification
  - Solver
  - CPU↔GPU
- Model Zoo
- GPU parallelism

```
name: "LogReg"
layer {
  name: "mnist"
  type: "Data"
  top: "data"
  top: "label"
  data_param {
    source: "input_leveldb"
    batch_size: 64
  }
}
layer {
  name: "ip"
  type: "InnerProduct"
  bottom: "data"
  top: "ip"
  inner_product_param {
    num_output: 2
  }
}
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "ip"
  bottom: "label"
  top: "loss"
}
```



SparkNet's Net API

Java Native Access (JNA) wrapper around Caffe

```
class Net {
    def Net(netParams: NetParams): Net
    def setTrainingData(data: Iterator[(NDArray,Int)])
    def setValidationData(data: Iterator[(NDArray,Int)])
    def train(numSteps: Int)
    def test(numSteps: Int): Float
    def setWeights(weights: WeightCollection)
    def getWeights(): WeightCollection
}
```

Example CNN NetParams

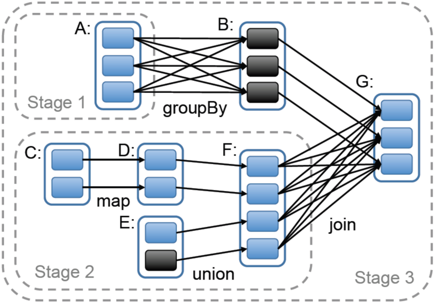
Compiled to protobuf Caffe network specification

```
val netParams = NetParams(
    RDDLayer("data", shape=List(batchsize, 1, 28, 28)),
    RDDLayer("label", shape=List(batchsize, 1)),
    ConvLayer("conv1", List("data"), kernel=(5,5), numFilters=20),
    PoolLayer("pool1", List("conv1"), pool=Max, kernel=(2,2), stride=(2,2)),
    ConvLayer("conv2", List("pool1"), kernel=(5,5), numFilters=50),
    PoolLayer("pool2", List("conv2"), pool=Max, kernel=(2,2), stride=(2,2)),
    LinearLayer("ip1", List("pool2"), numOutputs=500),
    ActivationLayer("relu1", List("ip1"), activation=ReLU),
    LinearLayer("ip2", List("relu1"), numOutputs=10),
    SoftmaxWithLoss("loss", List("ip2", "label"))
)
```

Spark [ZCD+12]

Resilient Distributed Datasets (RDDs)

1. *Lineage graph* describing lazy dataset
2. Aggressive in-memory caching (project Tungsten)
3. Partition-level dependency tracking

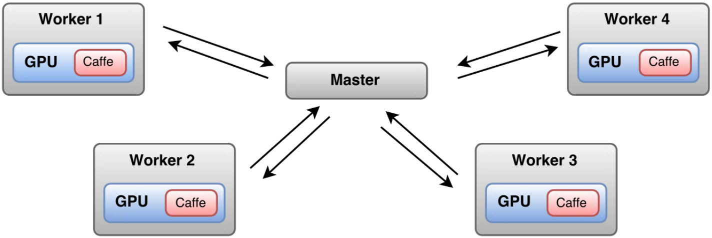


Parallelizing SGD

1. Master broadcasts model parameters to workers
2. Each worker runs SGD on its subset for $\tau = 50$ iterations
3. Worker model parameters collected and averaged on master

```
var trainData = loadData(...)
var trainData = preprocess(trainData).cache()
var nets = trainData.foreachPartition(data => {
    var net = Net(netParams)
    net.setTrainingData(data)
    net
})
var weights = initialWeights(...)
for (i <- 1 to 1000) {
    var broadcastWeights = broadcast(weights)
    nets.map(net => net.setWeights(broadcastWeights.value))
    weights = nets.map(net => {
        net.train(50)
        net.getWeights()
    }).mean() // an average of WeightCollection objects
}
```

SparkNet Architecture



Concerns in distributed optimization: communication

Inter-machine communication bottleneck

Worker machines must frequently read and write the global shared parameters.

Question: How does SparkNet's $\tau = 50$ affect communication? Convergence rate?

Concerns in distributed optimization: communication

Inter-machine communication bottleneck

Worker machines must frequently read and write the global shared parameters.

Question: How does SparkNet's $\tau = 50$ affect communication? Convergence rate?

- Communication once every τ instead of every single SGD iteration
- Workers obtain parameter estimates more biased to own partition

Concerns in distributed optimization: stragglers

The straggler problem

Many algorithms require synchronization barriers, which are rate-limited by the slowest machines.

Question: Why might some machines be slower than others?

Concerns in distributed optimization: stragglers

The straggler problem

Many algorithms require synchronization barriers, which are rate-limited by the slowest machines.

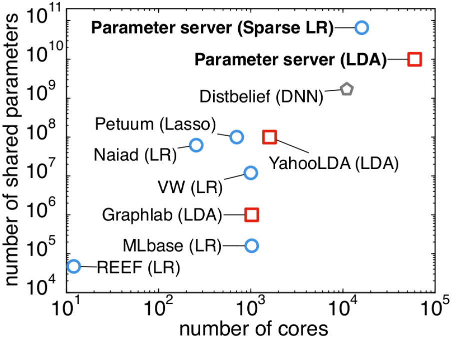
Question: Why might some machines be slower than others?

- Imbalanced workload partitioning
- Network congestion
- Other processes in multi-tenant environments

Addressing the straggler problem: asynchronous SGD

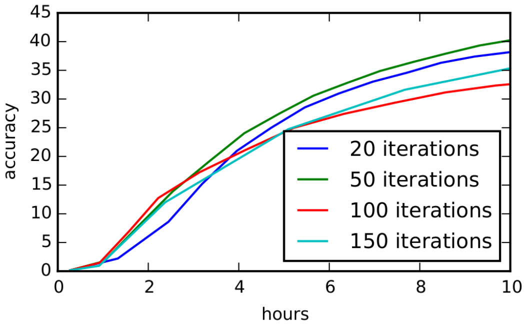
Not addressed by SparkNet, but an active research area:

- HogWild! – Correctness guaranteess [RRWN11]
- Parameter server – Distributed implementation [HCC+13], flexible consistency models in V3 [LZY+13]



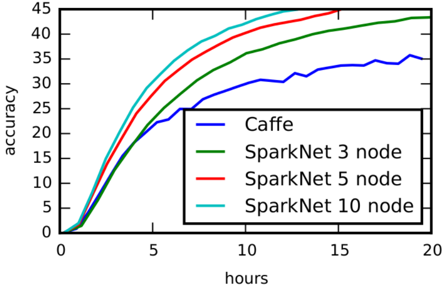
Tuning τ

- All experiments on EC2 g2.8xlarge nodes
- AlexNet [KSH12]
- $K = 5$ workers

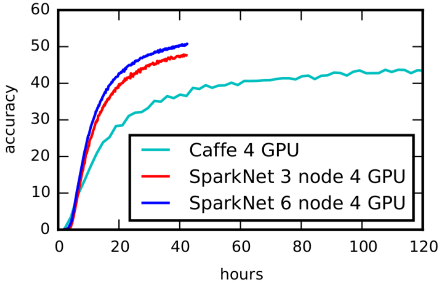


Training curves

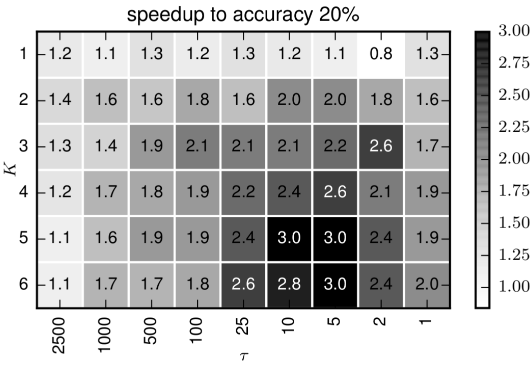
- AlexNet [KSH12]
- 8 layer CNN, ILSVRC2010
- $\tau = 50$



- GoogLeNet [SLJ+15]
- 22 layer CNN, ILSVRC2014
- $\tau = 50$



Speedups relative to single-GPU Caffe



- Why is $K = 1$ largely unaffected by τ?
- Why does performance deteriorate with increasing K? Increasing τ?

Related work

- Distbelief [DCM$^+$12]
  - Both data-parallel and **model-parallel** DNNs
  - Asynchronous (downpour SGD) and batch (sandblaster L-BFGS
  - Google Borg CPU cluster
- FireCaffe [IAMK15]
  - Caffe parameter server
  - Cray GPU cluster
  - **SoA** 47$\times$ **speedup** training GoogLeNet on 128 GPUs
- Yahoo/CaffeOnSpark [CNF16]
  - Infiniband/ethernet for GPU parameter exchange
  - **Spark and EC2 integration**

Conclusion

- Contributions
  - Data-parallel integration between a popular deep CNN tool (*Caffe*) and a general-purpose cluster computing (*Spark*) framework
  - Empirical studies on τ and K
- Criticisms
  - Does not solve straggler problem: synchronization barrier at each parameter collection step
  - No comparison against related work
  - Experiments are too small (10 nodes max)
- Future work
  - Mathematical analysis of τ's effects on learning convergence
  - Lock-free parameter updates (will require extending Spark)

References I

- Martın Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al., *Tensorflow: Large-scale machine learning on heterogeneous distributed systems*, arXiv preprint arXiv:1603.04467 (2016).
- Jun Shi Cyprien Noel and Andy Feng, *Large scale distributed deep learning on hadoop clusters*, 2015 (accessed May 17, 2016).
- Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al., *Large scale distributed deep networks*, Advances in Neural Information Processing Systems, 2012, pp. 1223–1231.

References II

- Jeffrey Dean and Sanjay Ghemawat, *Mapreduce: simplified data processing on large clusters*, Communications of the ACM **51** (2008), no. 1, 107–113.
- Seth Gilbert and Nancy Lynch, *Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services*, ACM SIGACT News **33** (2002), no. 2, 51–59.
- Qirong Ho, James Cipar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Phillip B Gibbons, Garth A Gibson, Greg Ganger, and Eric P Xing, *More effective distributed ml via a stale synchronous parallel parameter server*, Advances in neural information processing systems, 2013, pp. 1223–1231.

References III

- Forrest N Iandola, Khalid Ashraf, Mattthew W Moskewicz, and Kurt Keutzer, *Firecaffe: near-linear acceleration of deep neural network training on compute clusters*, arXiv preprint arXiv:1511.00175 (2015).
- Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell, *Caffe: Convolutional architecture for fast feature embedding*, Proceedings of the ACM International Conference on Multimedia, ACM, 2014, pp. 675–678.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, *Imagenet classification with deep convolutional neural networks*, Advances in neural information processing systems, 2012, pp. 1097–1105.

References IV

- Mu Li, Li Zhou, Zichao Yang, Aaron Li, Fei Xia, David G Andersen, and Alexander Smola, *Parameter server for distributed machine learning*, Big Learning NIPS Workshop, vol. 1, 2013.
- Philipp Moritz, Robert Nishihara, Ion Stoica, and Michael I Jordan, *Sparknet: Training deep networks in spark*, arXiv preprint arXiv:1511.06051 (2015).
- Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu, *Hogwild: A lock-free approach to parallelizing stochastic gradient descent*, Advances in Neural Information Processing Systems, 2011, pp. 693–701.

References V

- Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, *Going deeper with convolutions*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

- Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J Franklin, Scott Shenker, and Ion Stoica, *Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing*, Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation, USENIX Association, 2012, pp. 2–2.