Scalable Deep Learning in Baidu

Weide Zhang, Kyle Tsai, Jiang Wang
Baidu USDC

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO

Background

- Spark
  - Batch/Streaming processing, Spark SQL, MLLib
- Deep learning has many use cases in Baidu and showed significant improvement in quality
  - Image retrieval ranking
  - Ads CTR prediction
  - Machine translation
  - Speech recognition
- Goal: able to use distributed deep learning in Spark

Typical Training Data in Baidu

- Image Recognition: 100 millions
- OCR: 100 millions
- Speech: 10 billions
- CTR: 100 billions
- Grows every year since 2012

Baidu Spark One

Paddle

- Parallel Asynchronous Distributed Deep Learning Library
  - Support distributed parameter servers to do synchronous/asynchronous parameter update
  - Support Multi GPU / CPU training
  - Support sparse model
  - Easy to understand API for user to add new layers
  - Support rich features for deep learning use cases, especially for NLP

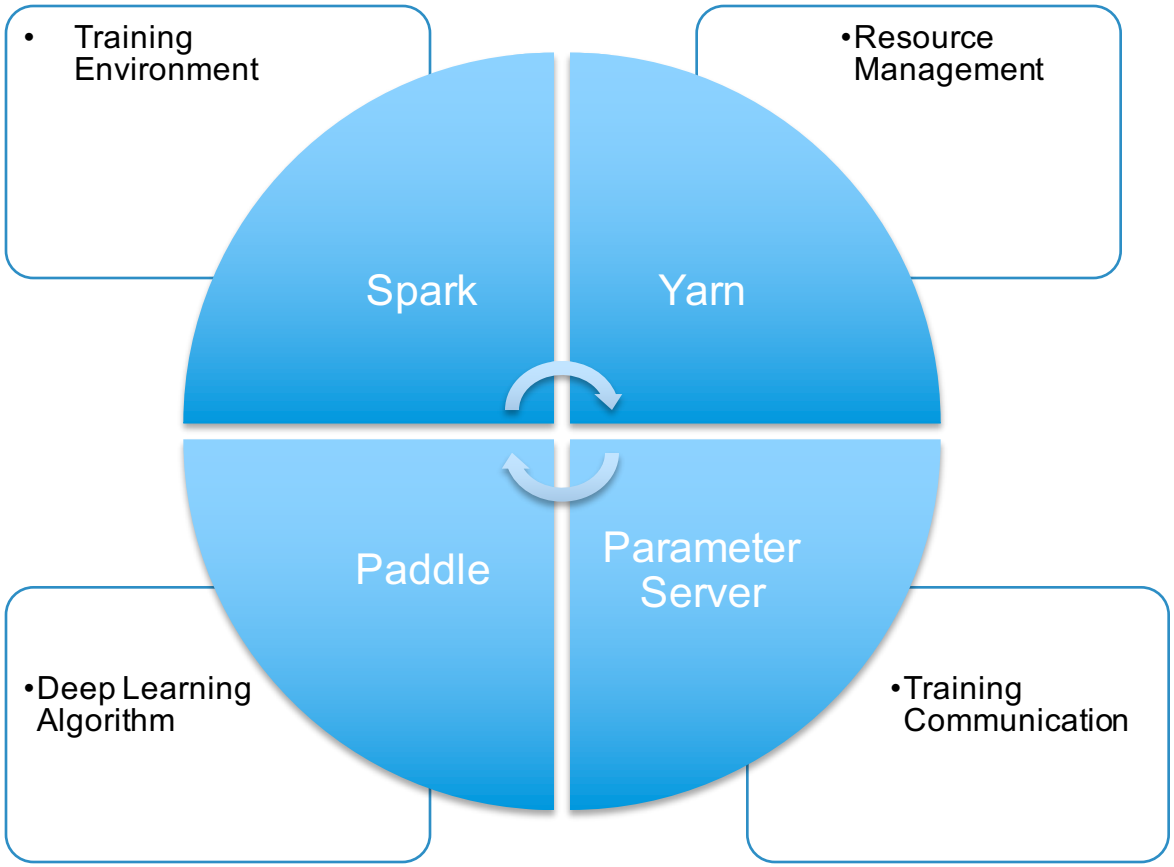
Deep learning options comparison

	Caffe	Tensor Flow	Torch	Paddle
Distributed Training	Yes	Yes	No	Yes
Communication Cost	Medium	High	N/A	Medium to Low
Easy to customize and coding	Yes	More Learning Curve	More Learning Curve	Yes
Sparse Model Support	No	Yes	Yes	Yes
Area of Focus	Vision	All	All	All
Integration with Spark	Yes	No	No	Yes

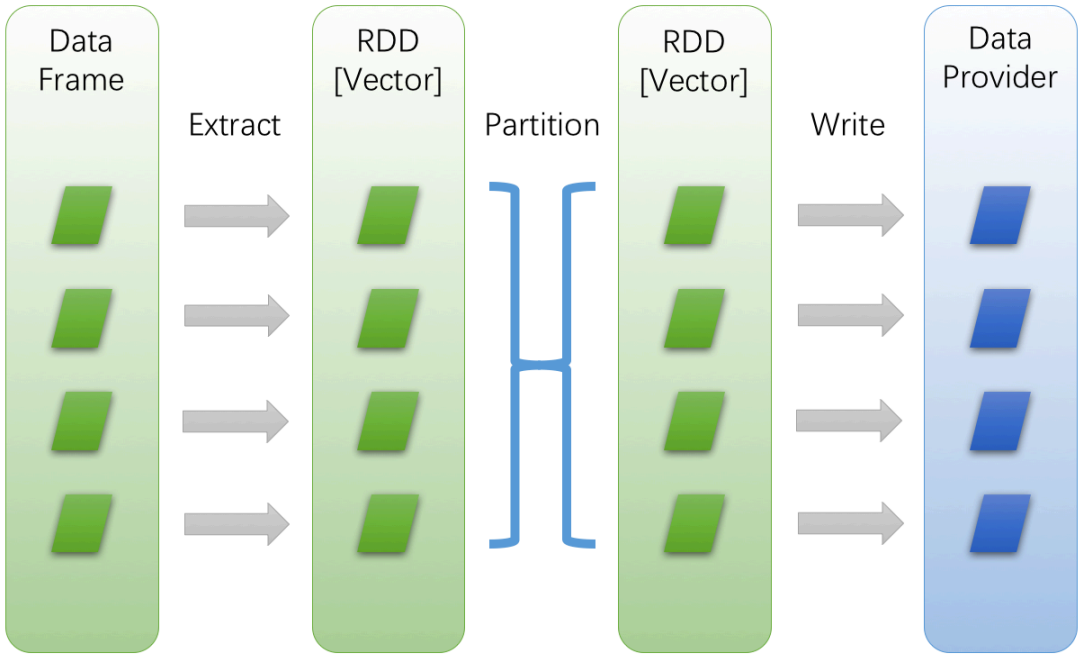
High Level Goals

- Implement Spark ML abstractions to let user train deep learning models with minimal code change
- Leverage paddle's native training and parameter server mechanisms to be scheduled in spark deep learning jobs
- Handle multi-tenancy and heterogeneity
- Parallelize hyper parameter selection
- Batch and Streaming learning

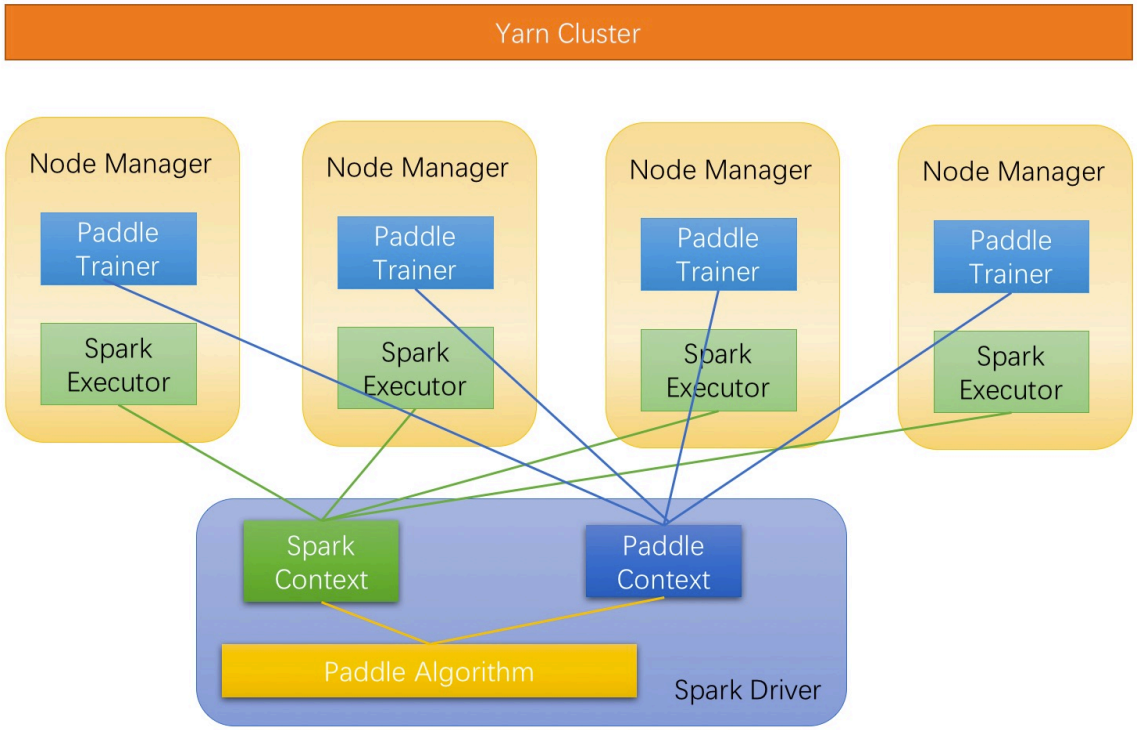
Paddle on Spark



Training Data Flow

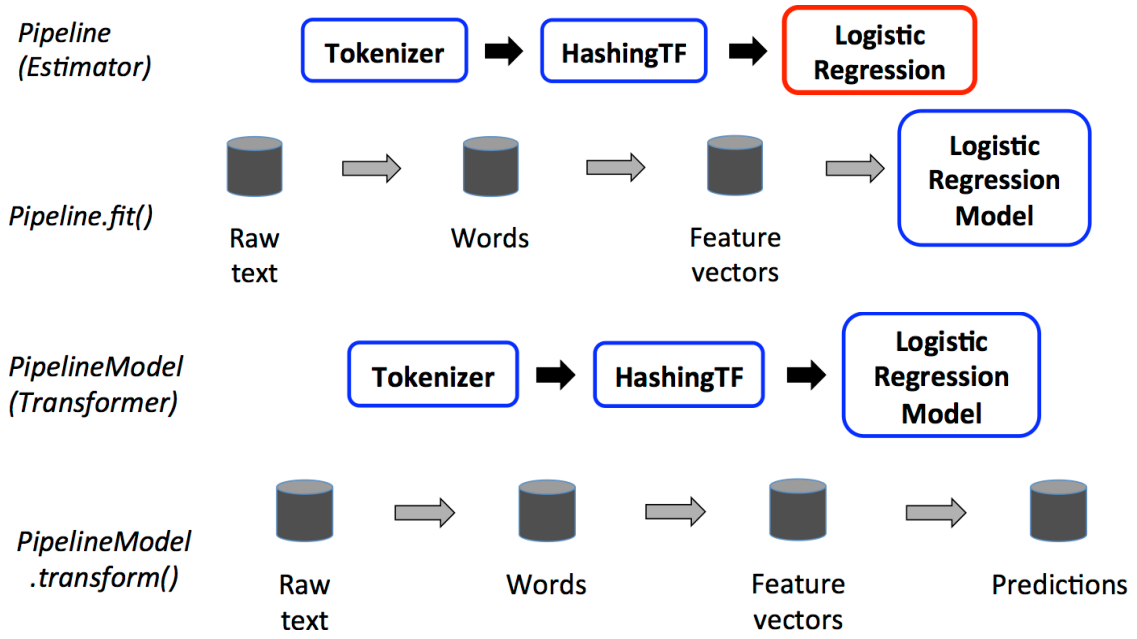


System Architecture

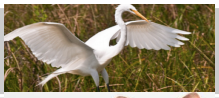


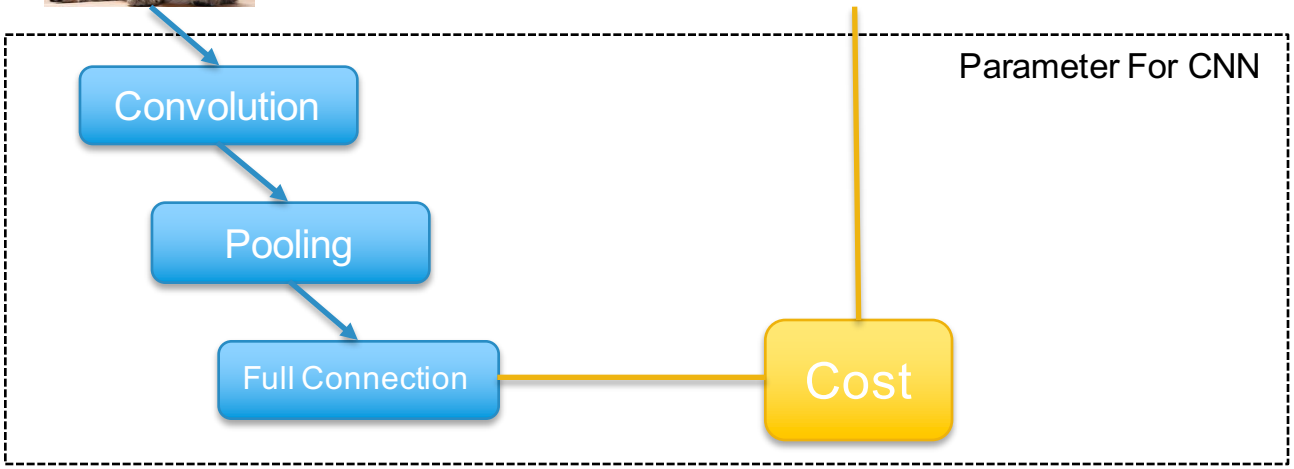
Spark ML's Abstraction

- Train
- Predict



Simple Parameter Is Not Enough

Image	Label
	Bird
	Cat



Use Paddle As Estimator

```
trait PaddleClassifierParams extends PredictorParams with NeuralNetworkParams

abstract class PaddleClassifier(override val uid: String) extends Predictor[Vector, Paddle, PaddleModel]
  with PaddleClassifierParams {

  override protected def train(input: DataFrame): PaddleClassifierModel
}

abstract class PaddleClassifierModel(override val uid: String) extends PredictionModel[Vector, PaddleModel] {
  override protected def predict(features: Vector): Double

  override def copy(extra: ParamMap): PaddleClassifierModel
}
```

Code your Configuration

```
def config(paddle: PaddleClassifier): Unit = {
  paddle.setFeaturesCol("image")
  paddle.setFeatureSize(224 * 224 * 3)

  paddle.setLabelCol("label")
  paddle.setLabelCount(1000)

  paddle.setLearningRate(0.02f / 128)
  paddle.setMomentum(0.9f)
  paddle.setDecayRate(0.0005f * 128)

  val conv = paddle.ConvolutionLayer("conv")
    .setActivationFunction(NeuralNetwork.RELU)
    .setNumFilters(8)
    .setPartialSum(110 * 110)
  conv.addInput(paddle.FEATURES)
    .setFilterSize(7)
    .setChannels(3)
    .setPadding(0)
    .setStride(2)
    .setGroups(1)
    .setInitialMean(0.0f)
    .setInitialStd(0.059f)
```

```
  val pool = paddle.PoolLayer("pool")
  pool.setInput(conv)
    .setPoolMethod(NeuralNetwork.MAX)
    .setChannels(8)
    .setSize(16)
    .setStride(8)

  val fc = paddle.FullConnectionLayer("fc")
    .setSize(1000)
    .setActivationFunction(NeuralNetwork.SOFTMAX)
  fc.addInput(pool)
    .setInitialMean(0f)
    .setInitialStd(0.001f)

  paddle.setOutputLayer(fc)
}
```

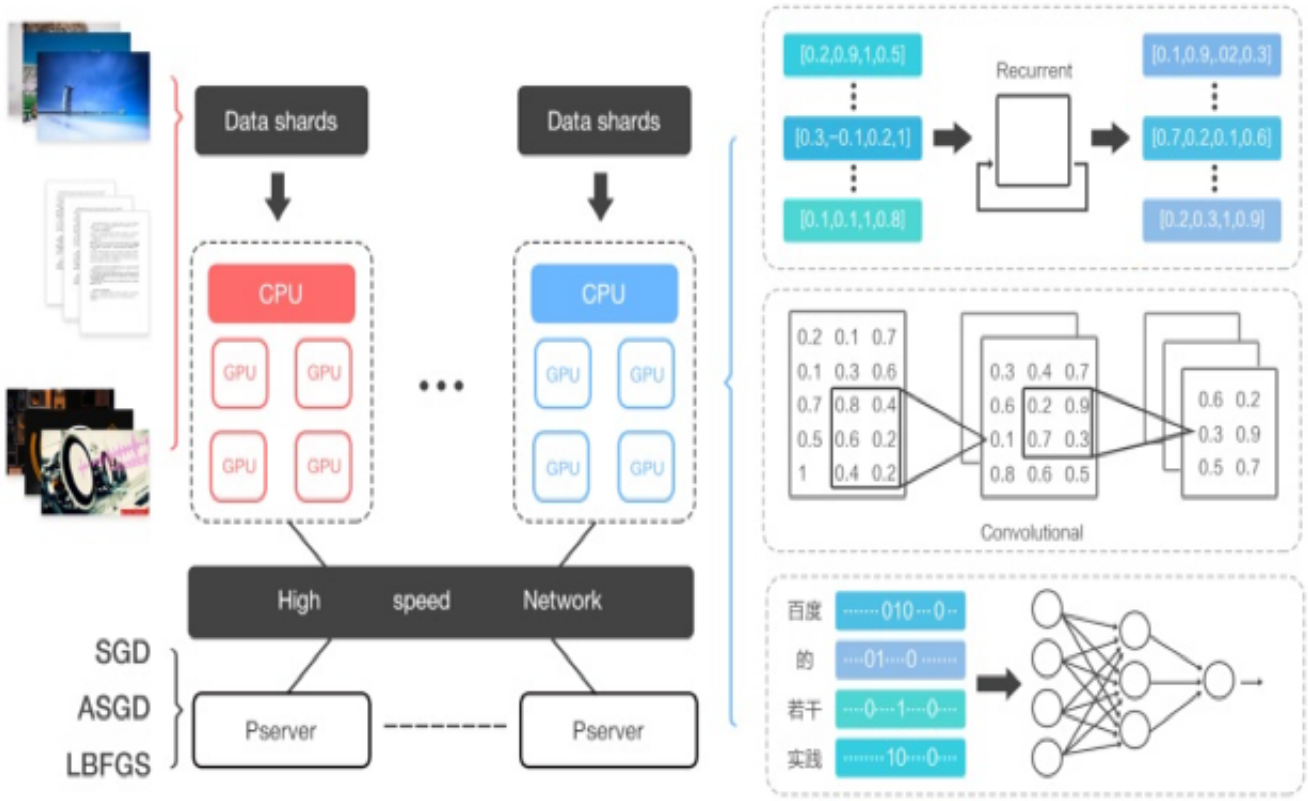
Example of caffe prototxt

```
name: "CaffeNet"
layer {
  name: "data"
  type: "Input"
  top: "data"
  input_param { shape: { dim: 10 dim: 3 dim: 227 dim: 227 } }
}
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  convolution_param {
    num_output: 96
    kernel_size: 11
    stride: 4
  }
}
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "conv1"
  top: "conv1"
}
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
```

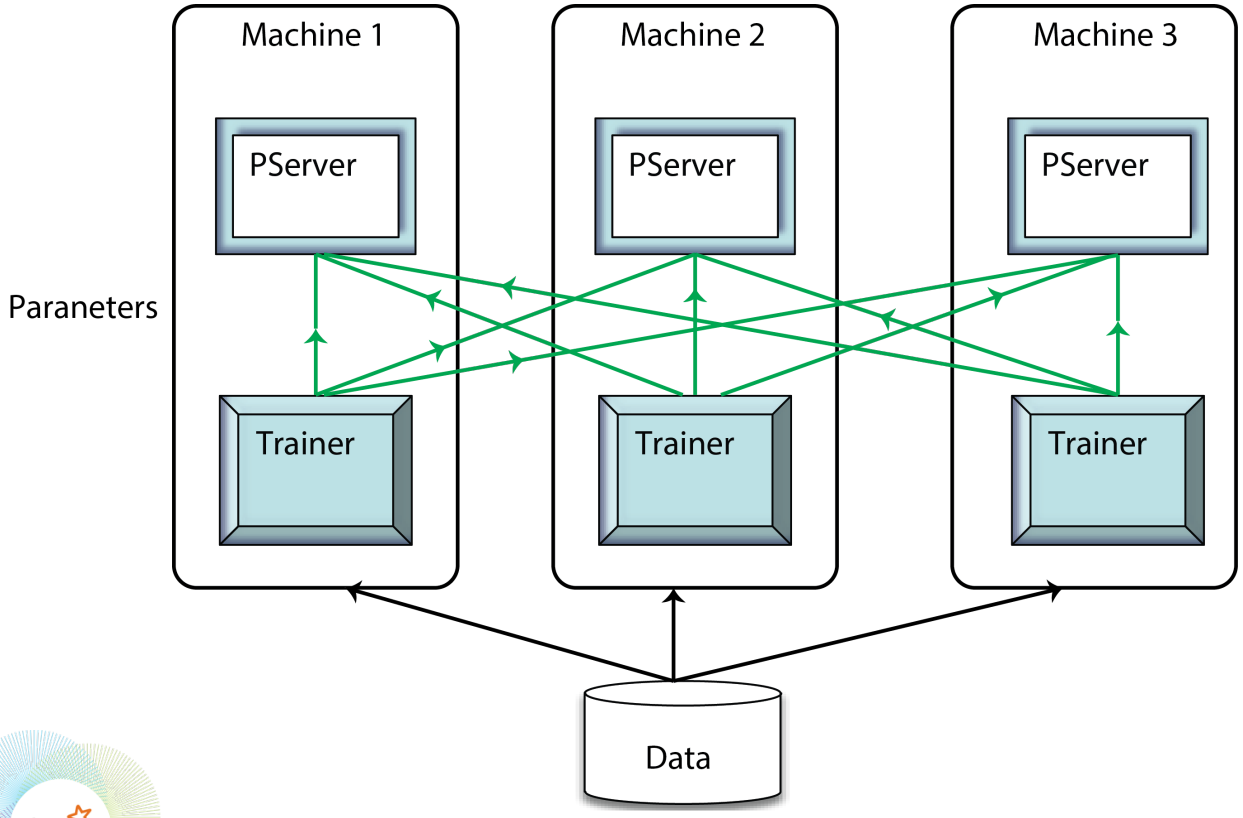
Design decisions

- Spark ML Compatible API
  - Compatible with Spark is more important than implemented under Spark
- Code level configuration
  - Easy and flexible
  - Manual is prone to error

PADDLE Scalable Deep Learning Platform at Baidu

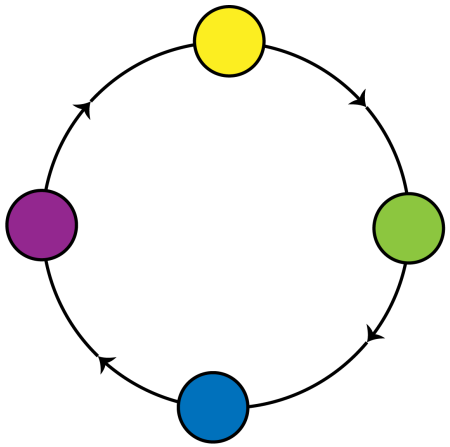


Sharded Parameter Server

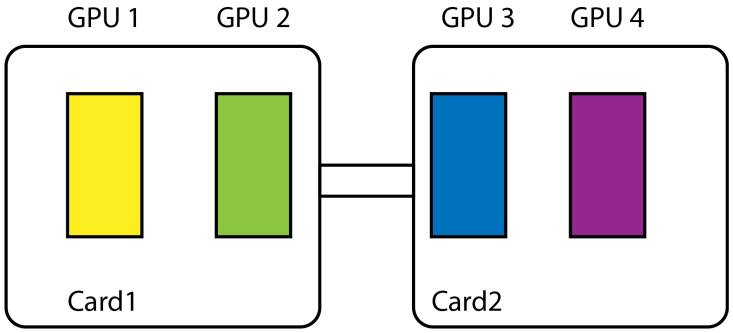


- One parameter an one trainer co-locate in a machine.
- Parameters are shared, but not replicated.
- All-to-all communication.
- Our environments
  - 4 GPUs per machine.
  - 4-10 machines.
  - all machines in one switch
  - reliable data center.

GPU Ring Synchronization



Ring Structure

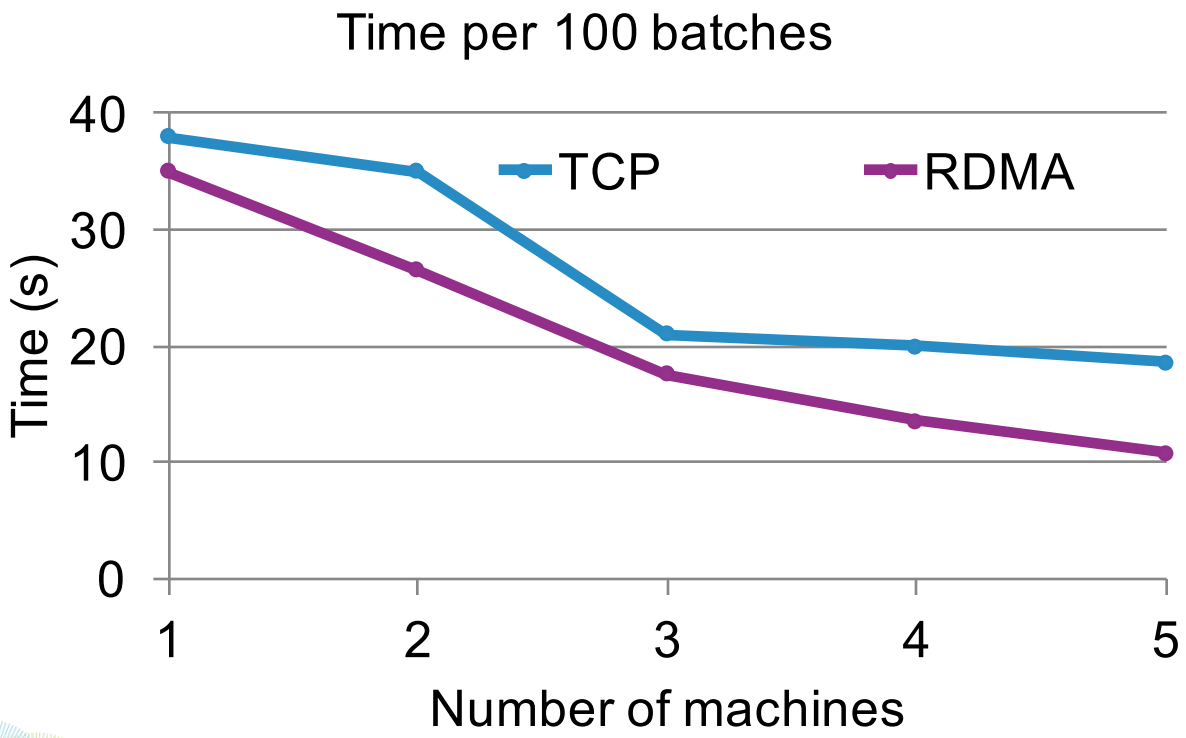


GPU Topology

Main parameter allocation

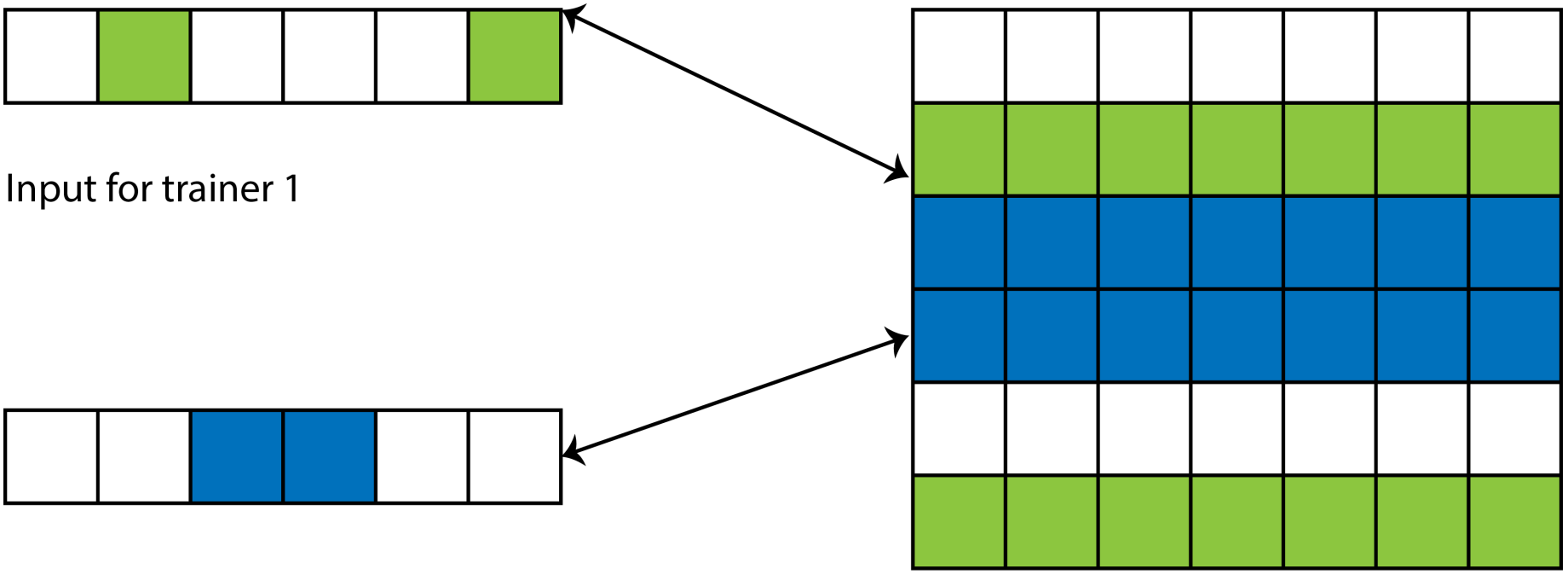
- Each parameter only needs to go through slow connection two times.
  - One for reduce.
  - Another for scatter.

ImageNet Scale Experiments

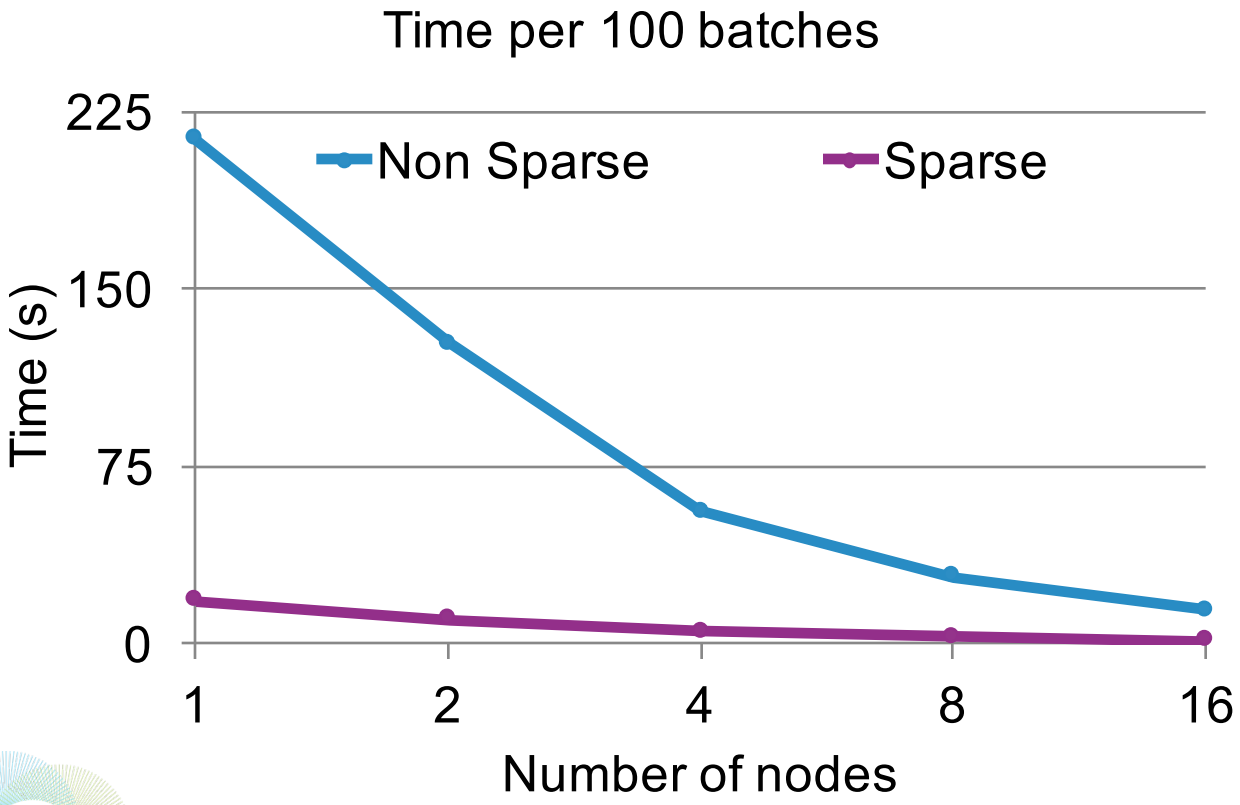


- AlexNet on ImageNet
- batch size = 64
- 1 Machine has 4 K10 GPUs.

Sparse Training

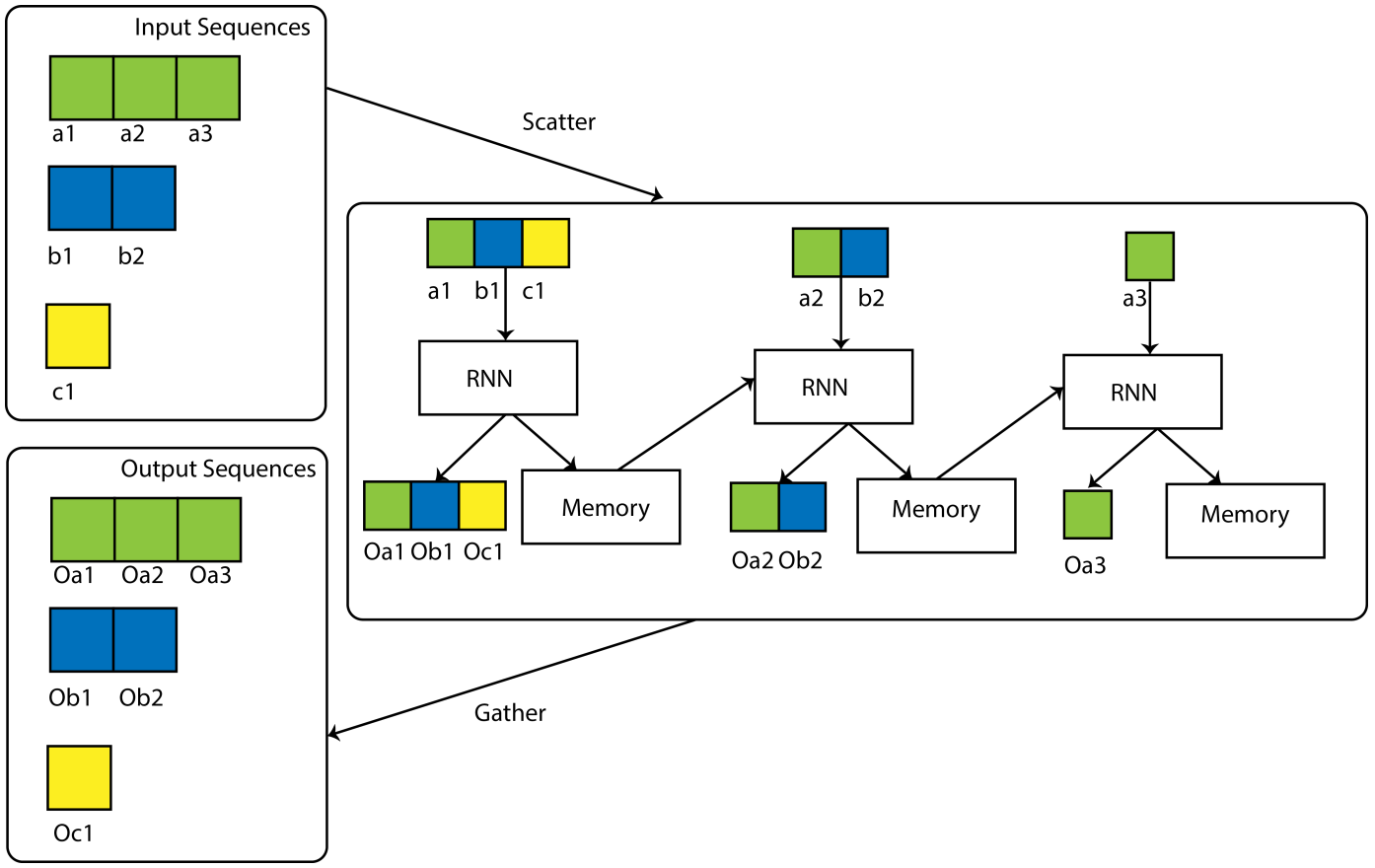


Sparse Training Experiment

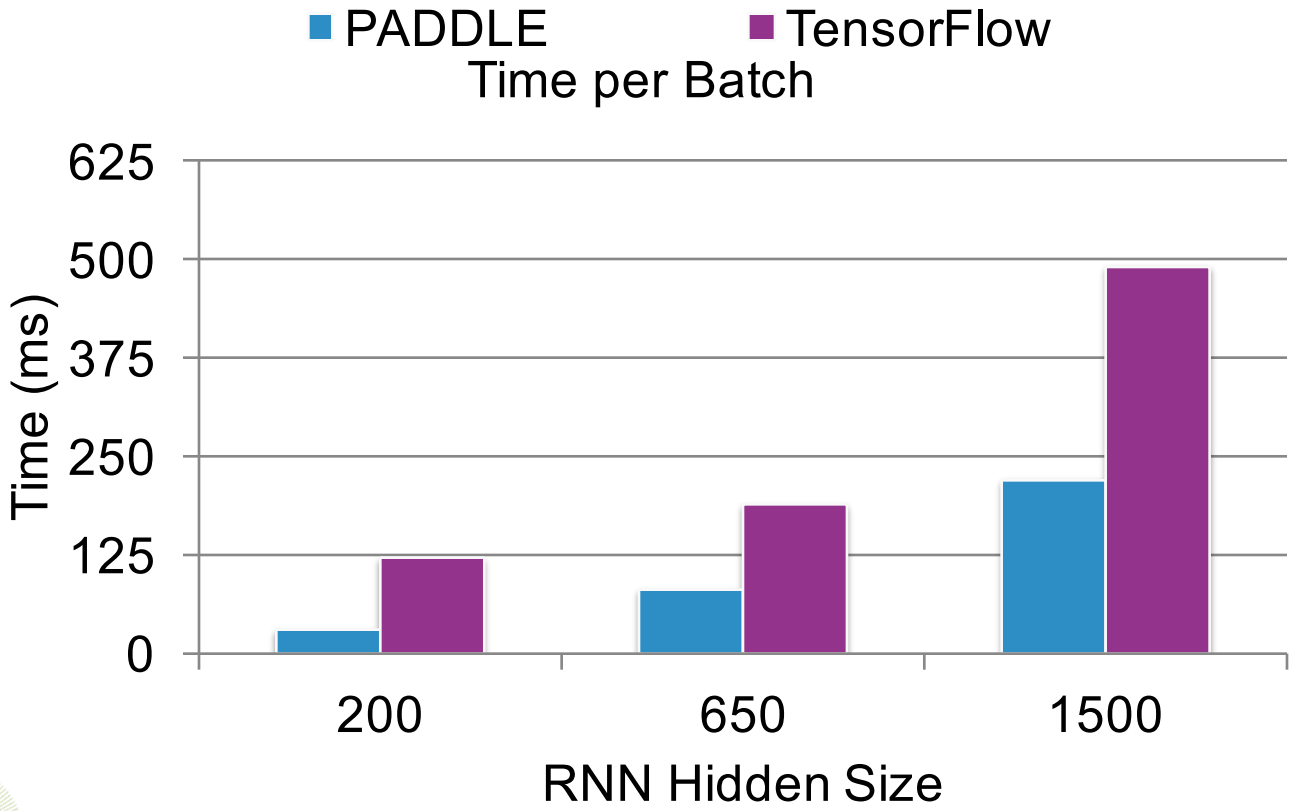


- 1451594 dimensional sparse feature.
- Embedded to 128d, 96d, 96d, and 128d.
- Using a ranking cost on the top.
- batch size = 128.

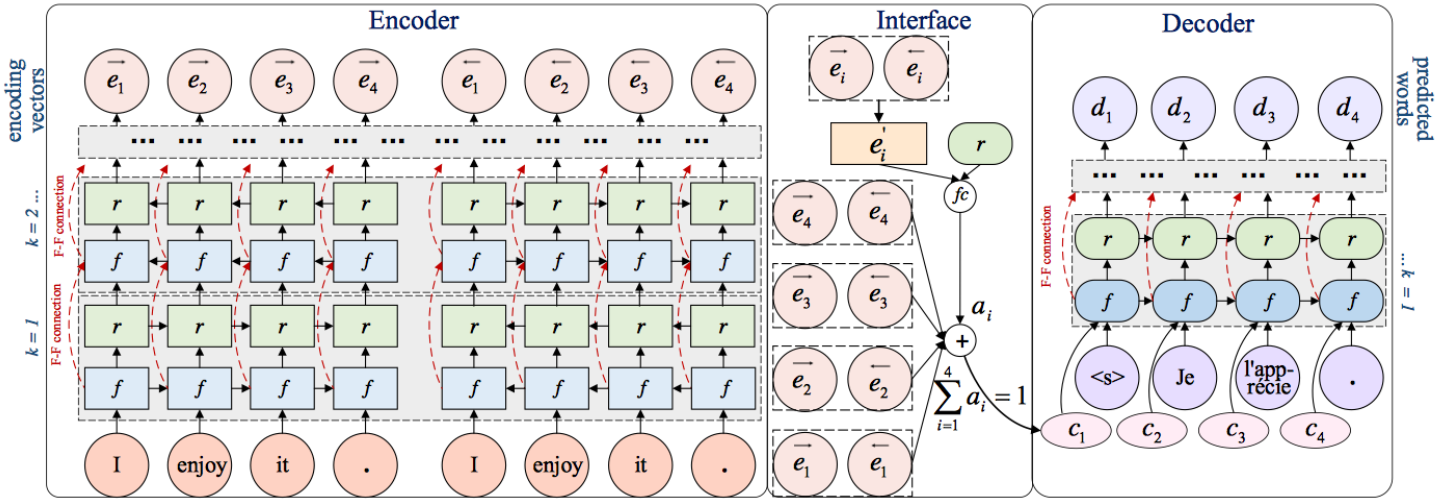
Flexible and Efficient RNN Implementation



RNN Performance Comparison with TensorFlow



- Machine Translation
- batch size = 64
- embedding size = hidden_size
- dictionary size = 10000



Distributed Training Performance Character Neural Machine Translation

- 8 Machines, each with 4 K40 GPUs
- number of RNN encoder layers: 9, number of RNN decoder layers: 7
- Word embedding size: 256, RNN size: 512
- batch size: 25000 character
- Speed:
  - attention: 25 minutes / 100 batches .
  - encoder-decoder: 9 minutes / 100 batches.

Future work

- Streaming training
- Dynamic trainer allocation
- FairScheduler
- Model serving

THANK YOU.

zhangweide/kyletsai/wangjiang03@baidu.com

SPARK SUMMIT 2016
DATA SCIENCE AND ENGINEERING AT SCALE
JUNE 6-8, 2016 SAN FRANCISCO