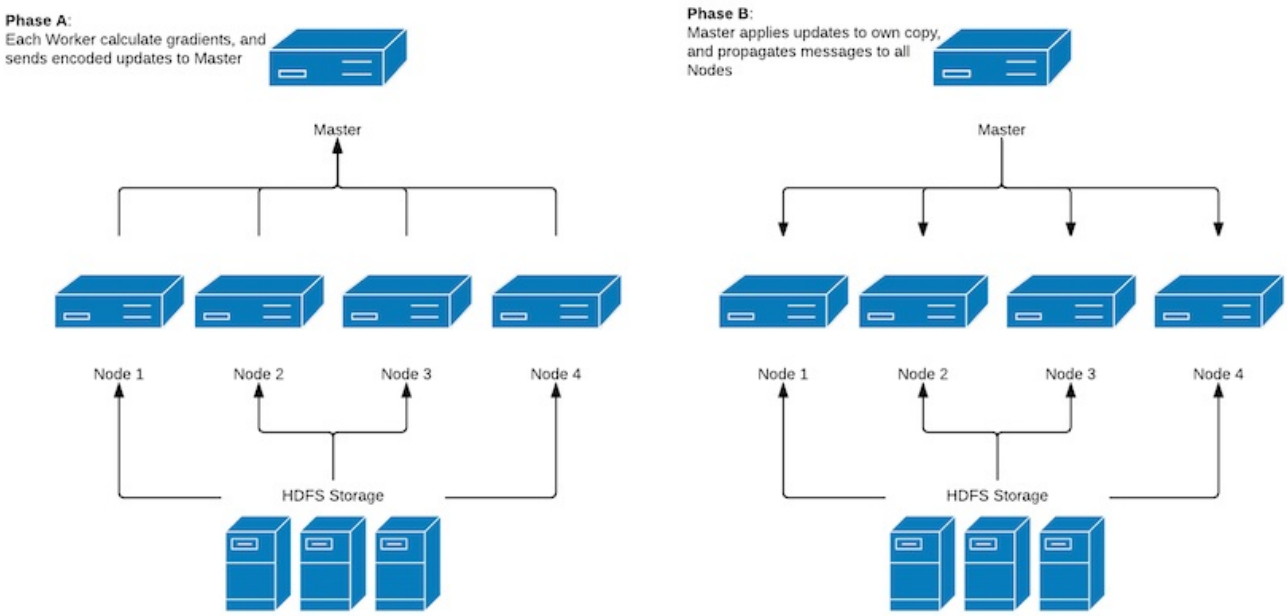
Deeplearning4j分布式训练：梯度共享

0.9.1版本（或0.9.2-SNAPSHOT）起，DeepLearning4j开始支持Apache Spark环境中的分布式训练，同时支持在Spark环境外用Aeron实现高性能节点间通信。

分布式训练的核心思想很简单：每个工作器根据自己的数据集计算梯度。
在梯度被用于调整网络权重之前，它们会首先在一个中间存储机制中（每台计算机有一个）累积起来。
聚合之后，高于特定阈值（可设置）的更新值以稀疏二元数组的形式传播给整个网络。
低于该阈值的更新值则被存储起来，与之后的更新累加，因此不会丢失，只是通信延迟。
与整体发送（稠密的）更新或参数向量的不成熟方法相比，这一设置阈值的方法可以将网络通信需求降低许多个数量级， 同时保持较高的准确率。 更多有关阈值法的详情请参见Strom，2015：《基于商用GPU云计算的可扩展分布式深度神经网络训练（Scalable Distributed DNN Training using Commodity GPU Cloud Computing）》和《分布式深度学习，第1部分：分布式神经网络训练简介》。

Nikko Strom 在原始算法中增加了一些额外的参数：

- 变量阈值（Variable threshold）：如果每次迭代的更新次数太少，阈值会自动降低一个可配置的step值。
- 稠密位图编码（Dense bitmap encoding）：如果每次迭代的更新次数太多，则使用另一种编码方案，其为任何给定的更新消息提供通过线路发送的“最大字节数”的保证。
- 定期地，我们发送“摇动（shake up）”消息，编码一个明显较小的阈值，以共享延迟的权重，编码阈值不能超过当前的阈值。



请注意，使用Spark确实会产生系统开销。为了确定Spark对您是否有利，请考虑用性能侦听器（Performance Listener）检查以毫秒为单位的迭代时间。 如果迭代时间不长于150ms，那么可能不值得使用Spark。

建立集群

运行训练任务仅需要Spark 1.x/2.x集群和至少一个开放的UDP端口（同时允许入站/出站）。

集群设置

如上文所述，DeepLearning4j支持Spark 1.x和Spark 2.x集群。但是，运行这一特定的实现版本还需要Java 8+。如果您的集群运行的是Java 7，您需要升级Java或者采用参数平均化训练模式。

网络环境

梯度共享很大程度上依靠UDP协议来确保主节点和工作节点之间在训练时的通信。如果您在AWS或Azure这样的云环境中运行集群，您需要为入站/出站连接开放至少一个UDP端口，并且在传递`给SharedTrainingMaster`构建器的`VoidConfiguration.unicastPort(int)`bean

中指定该端口。

需要记住的另一个选项是，在使用YARN（或其他任何处理Spark网络连接的资源管理器）的情况下，您必须为网络指定将被用于UDP通信的网络掩膜。具体做法可以如下：`VoidConfiguration.setNetworkMask("10.1.1.0/24")`。

IP地址选择的最后手段是`DL4J_VOID_IP`环境变量。在您运行的每个节点上将这一变量设为通信所要使用的本地IP地址。

网络掩码

网络掩码是CIDR表示法，只是一种告诉软件应该使用哪个网络接口进行通信的方法。假如你的机器里面有三个主机被分配了如下的三个IP地址：`192.168.1.23, 192.168.1.78, 192.168.2.133`，三个网络地址中公共的部分为`192.168.*`，所以网络地址掩码为`192.168.0.0/16`。你也可以在维基百科中获得更加详尽的解释：https://en.wikipedia.org/wiki/Subnetwork
我们使用网络掩码主要是应用于：Spark集群运行在Hadoop集群之上，或者是其他不承担Spark IP地址的环境（这里翻译起来很怪，希望有人能给我纠正一下）。在这种情况下，应该在VoidConfiguration bean中提供有效的网络掩码，并且将用它来为Spark外部通信选择接口。

依赖项

以下是唯一一个必需依赖项的模板：

```
<dependency>
    <groupId>org.deeplearning4j</groupId>
    <artifactId>dl4j-spark-parameterserver_${scala.binary.version}</artifactId>
    <version>${dl4j.spark.version}</version>
</dependency>
```

在示例中：

```
<dependency>
    <groupId>org.deeplearning4j</groupId>
    <artifactId>dl4j-spark-parameterserver_2.11</artifactId>
    <version>0.9.1_spark_2</version>
</dependency>
```

配置示例：

以下是从此处的一个示例项目中摘取的代码片段：

```
SparkConf sparkConf = new SparkConf();
sparkConf.setAppName("DL4J Spark Example");
JavaSparkContext sc = new JavaSparkContext(sparkConf);

MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
            .seed(12345)

.optimizationAlgo(OptimizationAlgorithm.STOCHASTIC_GRADIENT_DESCENT)
            .iterations(1)
            ...
            .build();

/*
    这是ParameterServer配置的bean。实际会用到的选项只有.unicastPort(int)
*/
VoidConfiguration voidConfiguration = VoidConfiguration.builder()
            .unicastPort(40123)
            .build();

/*
    SharedTrainingMaster是分布式训练的基础。它包含训练所需的所有逻辑。
*/
TrainingMaster tm = new SharedTrainingMaster.Builder(voidConfiguration
,batchSizePerWorker)
            .updatesThreshold(1e-3)
            .rddTrainingApproach(RDDTrainingApproach.Export)
            .batchSizePerWorker(batchSizePerWorker)
            .workersPerNode(4)
            .build();

//创建Spark网络
```

```
30.  SparkDl4jMultiLayer sparkNet = new SparkDl4jMultiLayer(sc, conf, tm);
31.
32.  //执行训练：
33.  for (int i = 0; i < numEpochs; i++) {
34.      sparkNet.fit(trainData);
35.      log.info("Completed Epoch {}", i);
36.  }
```

请注意：上述配置假定集群中所有节点上都已经打开了UDP端口40123。

有效的可扩展性

网络IO有自己的代价，而这一算法也可以进行一些IO。额外训练时间开销的计算方法为`更新编码时间 + 消息序列化时间 + 应用来自其他工作器的更新`。 最初的迭代时间越长，共享产生的相对影响就越小，得到的理论扩展性就越好。
下面这个简单的表格可以帮助您估计可扩展性：

迭代时间	编码时间	解码时间	更新时间	服务开销
550	50	5	50	20

节点数量	每个节点的工作器数量
8	4

可扩展性：70.51%

性能提示

执行器、处理器核、并行计算

Spark的设计让您可以为任务设置执行器数量以及每个执行器所用的核的数量。假设您的集群中有18个节点，每个节点有32个核。 在这种情况下，`--num-executors`值等于18，而推荐的`--executor-cores`值可以在2到32之间。这一选项实质上是在指定将RDD划分为多少个分区。 此外您还可以手动设定每个节点所用的DL4J工作器的具体数量。这可以用`SharedTrainingMaster.Builder().workersPerNode(int)`方法来实现。

如果您的节点以GPU驱动，一般最好将`workersPerNode(int)`设为每台主机的GPU数量，或者保持默认值以供自动调试。

编码阈值

阈值较高时，更新较稀疏，网络IO性能将会提高，但有可能（很可能会发生）影响神经网络的学习效果。 阈值较低时，更新更稠密，每次更新的消息会变得更大。网络IO性能将因此降低。我们无法预测具体情况下的“最佳阈值”，因为不同架构下的最佳阈值区别很大，但不妨先从`1e-3`的默认值开始尝试。

网络延迟与带宽

基本法则很简单：网络越快，性能越好。现如今，1GBe的网络会被视为最低的配置，但由于延迟较小，10GBe网络的适用性更好。 当然，性能还取决于网络规模和计算量。网络越大，所需的带宽越大，但每次迭代需要的时间更长（因此可能为异步通信留下更多时间）。

UDP单播与UDP广播

为了确保最大化的兼容性（例如，AWS和Azure这样的云计算环境不支持多播），目前DL4J只采用UDP单播。UDP广播传输应会更快，但对于训练表现而言，这种差异应当可以忽略（除非工作量非常小）。按照设计，每个工作器每次迭代发送一条更新消息，无论UDP传输类型如何，这一点都不会改变。UDP单播传输中，消息的重新传送是由主节点处理的（主节点通常利用率较低），而消息的传递是异步进行的，因此我们在性能方面只要求更新的通信时间低于网络迭代时间，而实际情况也通常如此。

多GPU环境

我们预计支持设备间PCIe/NVLink P2P连接的主机性能会最好。但是，整个系统即使没有P2P传输也能正常运作。只是会慢“一点点”而已。:)

原文地址：https://deeplearning4j.org/distributed
更多文档可以查看 https://github.com/sjsdfg/deeplearning4j-issues。
欢迎star