目录

4 可行性分析

前期的调研报告提出，可以从存储安排优化和文件预存取两方面来改进海量小文件的存取问题。以下主要讨论文件预存取的可行性。

4.1 理论依据

4.1.1 分布式文件系统存取文件的流程

以 Hadoop 为例，其存取文件的流程如下：

写文件

- client 向 namenode 发送创建文件的请求。
- namenode 检测待创建的文件是否存在，如果可以创建，则返回待写入的 datanode 的地址。
- client 向 datanode 中写入文件，当一个 datanode 中的 block 写入完成后，会异

步地在其他的 datanode 中的 block 进行复制。

- datanode 之间、datanode 和 client、client 和 namenode 之间反馈上传完成的确认信息

读文件

- client 向 namenode 发送读取文件的请求
- namenode 返回待读取文件所在的 datanode 的存储地址
- client 从 datanode 中读取文件

单从存取文件的流程来看，似乎存取文件的瓶颈是网络带宽，但这其实只是对大文件而言的。对于海量的小文件，存取速度则会受限于硬盘。我们有这样的经验，当在电脑上删除大量的小文件时，速度会变得非常慢，甚至于只有每秒几 K。这是因为尽管文件总大小可能不大，但是文件数量过多，磁盘磁头需要不断寻道，然后才对文件进行操作，真正删除文件所用时间并没有占多少。对于分布式文件系统也是一样，需要在短时间内找到大量小文件的地址，性能可想而知。

一个比较粗暴的解决办法就是，提升硬件性能，也就是换一块性能更好的硬盘，比如固态硬盘。但这其实并不会有太大的改善。有人曾经做过测试，向固态硬盘中拷贝 100 万个大小只有几十字节的小文件，文件总大小是几十兆，但拷贝速度仍然只有每秒几十 K。

另一种办法是，提前读取或写入用户（或者应用等等）可能存取的文件，也就是预测用户（或者应用）的行为，进行预存取。这样就避免接触到硬件上的瓶颈，在尽可能不改变 DFS 架构和硬件配置的情况下提升性能。这正是我们要采取的办法。

4.1.2 几个通过预测提高性能的案例

CPU 分支预测(branch prediction)

CPU 通过流水线技术来提高效率。所谓流水线，就是将一条指令的执行分成几个阶段，在同一时刻执行不同指令的不同阶段，以达到提高吞吐率的目的。

但是当执行到一个分支跳转语句的时候，由于不知道下一条指令究竟是什么，流水线不得不阻塞，一直到分支跳转语句执行完，确定下一条指令的地址才能继续。这样就会降低速度。

分支预测就是为了解决这个问题而提出的，它预测分支的结果并立即沿预测方向执行。如果分支预测正确，那么显然就节省了等待分支指令执行的时间，如果预测错误，就会导致流水线的停顿，比等待分支指令执行完产生额外的时间开销，因为需要将已经填入流水线的指令清除。因此，提高分支预测的准确率，可以有效提高 CPU 的效率。

内存预取

Windows XP：Prefetch，即预取，但这里的预取比较低级，它只是在某个程序运行时，记录下这个程序经常从硬盘读取的文件。当下一次启动一个程序时，系统会到那些记录中查找有无关于这个程序的记录，如果有，则预先将那些可能需要用到的文件载入内存，然后才载入该程序。可以看出，这里的预取只有在程序启动时才会进行，之前仅仅是记录以下而已。

Windows Vista 及以后：Superfetch，这比 Prefetch 要高级不少，它会记录用户的使用习惯（如经常在某个时刻启动某个程序），然后自动预先（不同于 Prefetch 的被动预先）将硬盘中的文件载入内存。这个和我们将要采用的解决 DFS 小文件存取问题的策略很类似。

文件索引

打开 Windows 操作系统的任务管理器，有时可能会看到 SearchProtocolHost 这个进程，它的作用是：在计算机空闲的时候自动索引硬盘中的某些文件，这样，当我们需要搜索某个文件时，能够较快地给出结果，改善用户的体验。这其实也是一种预取策略。

页表

4.2 技术依据

4.1 节中给出了通过预测来提升 DFS 的海量小文件存取性能的理论可行性，下面主要介绍神经网络以及神经网络如何用于预测。

4.2.1 神经网络简介

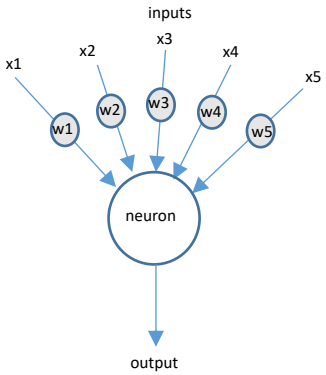
人工神经网络（Artificial Neural Networks，ANNs），简称神经网络，顾名思义，它是模仿动物大脑的神经网络的行为特征而提出的一种计算模型，其根本目的在于处理各种信息。

基本结构

动物大脑的神经网络是由一个个神经元通过突触相互连接而成的，人工神经网络中自然也有相似的结构。

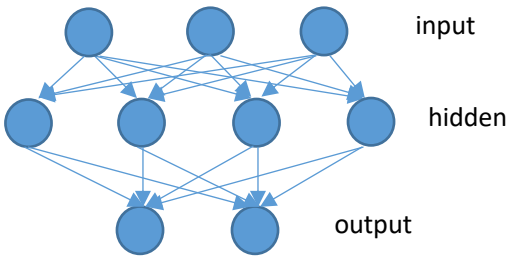
图 1 是一个人工神经细胞。x_1, x_2, x_3, x_4, x_5是神经细胞的输入信号，w_1, w_2, w_3, w_4, w_5是相应的输入信号的权重，反映了该信号对神经细胞状态的影响程度，由此可以得到对于该神经细胞的激励值$\Sigma_{i=1}^{5} w_i x_i$，然后再通过一个激活函数$f(x)$，将激励值映射成神经细胞的输出值$f\left(\Sigma_{i=1}^{5} w_i x_i\right)$。

常用的激活函数有 sigmoid 函数、tanh 函数、ReLU（rectified linear units）函数等等。



图一 人工神经细胞

通过多个这样的神经细胞，就可以组成一个神经网络。最简单的一种神经网络是前馈网络（feed-forward network），它由输入层（input layer）、隐藏层（hidden layer）和输出层（output layer）这三部分组成，其中隐藏层可以有任意多层。每一层的神经细胞只与前一层相连，信息只会向前传递，而同一层的神经细胞没有连接，如图 2。



图二 一个单层前向反馈神经网络

当然还有更复杂的神经网络，如深度神经网络（Deep Neural Network，DNN）、卷积神经网络（Convolutional Neural Network，CNN）、递归神经网络（Recursive Neural Network，RNN）等等。下一节中将详细介绍 RNN，这是我们采用的预测模型。

训练算法

所谓神经网络的训练，就是通过给定一组训练集（training set），根据神经网络的实际输出，不断调整输入信号的权重，使输出结果与期望输出之间的误差达到最小。从数学的角度来讲，神经网络的训练就是求损失函数（loss function，可以粗略理解为实际输出与期望输出之差）的最小值。由此可以产生以下几种训练算法：

- 梯度下降算法
- 牛顿算法
- 共轭梯度法

应用领域

不同结构的神经网络适用于不同的领域。

深度神经网络（DNN）：顾名思义，DNN 可以理解为有很多隐藏层的神经网络，

卷积神经网络（CNN）：

递归神经网络（RNN）：

值得注意的是，这些神经网络的分类并没有严格的界限，往往是相互交叉的。

4.2.2 循环神经网络

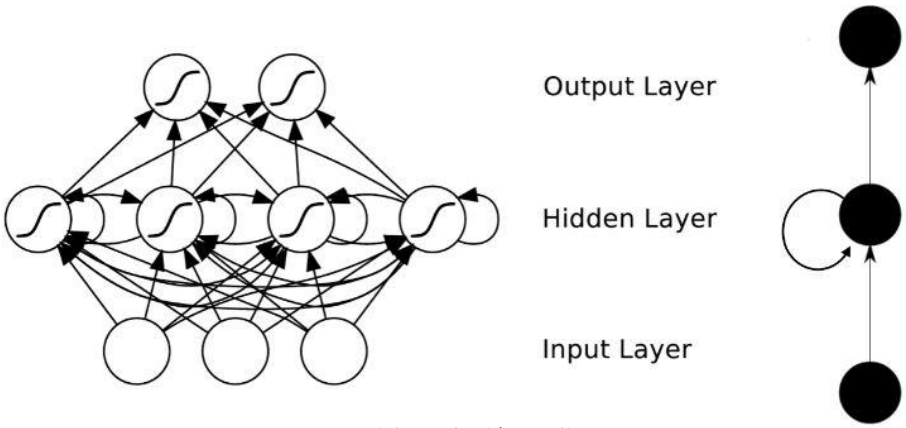
递归神经网络（Recursive Neural Network，RNN）可分为结构递归神经网络（Recursive Neural Network，RNN）和时间递归神经网络（Recurrent Neural Network，RNN，即循环神经网络），通俗地讲，结构递归神经网络可理解为空间上的循环，而循环神经网络则是时间上的循环。

我们要根据用户已有的存取文件的行为来预测用户将来可能存取的文件，这些已经存在的行为可以看成一个时间上相关的序列，要根据已有的序列来预测序列的下一个是什么，就好像根据一个句子来预测句子的下一个单词。传统的神经网络对此无能为力，因为一个句子的前后单词不是独立的，预测下一个单词是什么需要用到之前的信息，这就需要神经网络对之前的信息进行记忆，因而用到循环神经网络。这似乎有点像时序逻辑电路和组合逻辑电路的区别。

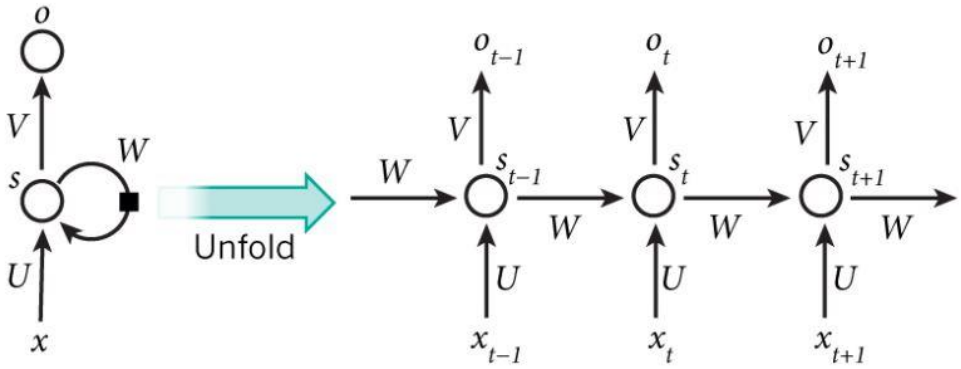
基本结构

图三是一个循环神经网络，可以看出，神经细胞与同一层的其他神经细胞及自身都有连接，这种结构使得神经网络可以记忆上一个时刻的输出，并影响到现在的输出。

如果我们将循环神经网络按照时间顺序展开的话，如图四，就会发现，这其实是一种链式结构，这也暗示了循环神经网络可以用来处理序列相关的问题。



图三 循环神经网络



图四 循环神经网络的展开

缺陷

从理论上来讲，循环神经网络可以处理任意长度的序列，但其实还有很多其他依赖因素。

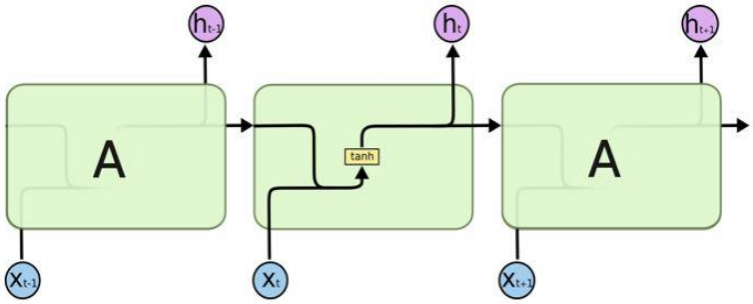
举一个简单的例子：The clouds are in the sky.循环神经网络可以很容易就预测出最后一个单词是 sky，因为它距离预测这个单词需要的信息 clouds 很近。而对于另一个句子：The cat, which already ate a bunch of food, was full.预测单词 was 则会有一定的困难，因为预测这个单词需要的信息 cat 离得太远了。这就是所谓的长期依赖问题（Long-Term Dependencies），当预测位置与预测所需要的相关信息的间隔太远时，循环神经网络会失去学习连接到那么远的信息的能力。

4.2.3 长短期记忆网络

简介

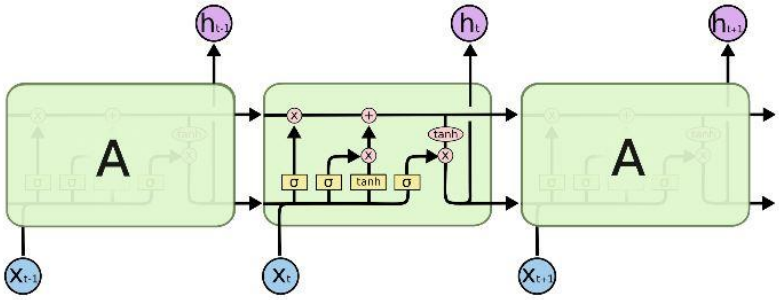
长短期记忆（Long Short-Term Memory，LSTM）网络由 Hochreiter 和 Schmidhuber (1997) 提出，它解决了 RNN 的长期依赖问题。

LSTM 是一种特殊的循环神经网络，它只是对 RNN 的隐藏层做了修改。



The repeating module in a standard RNN contains a single layer.

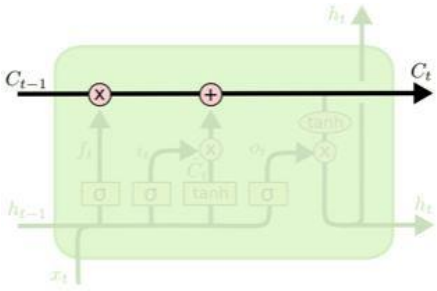
图五 一般 RNN 的重复模块



The repeating module in an LSTM contains four interacting layers.

图六 LSTM 的重复模块

如图五、六，一般的循环神经网络的（时间上）重复模块只有一个层，而 LSTM 的重复模块则有四个层，且这四个层以一种特殊的方式相互连接。



图七 神经细胞的状态控制

LSTM 的关键在于其神经细胞状态的流传，如图七。神经细胞的状态就在这样一条时间链上前进，每个时刻决定保留、更新或丢弃细胞状态的某些信息，而这个决定是由三个门来控制的。

神经细胞状态的控制

- 确定丢弃哪些信息：通过忘记门层（forget gate layer）完成，该门会读取此时的输入x_t和上一个时刻的输出h_{t-1}，通过$sigmoid$激活函数向细胞状态C_{t-1}中的每个数字输出一个 0 到 1 之间的数值，1 表示保留，0 表示丢弃。
- 确定更新哪些信息：分为两步，首先通过输入门层（input gate layer）（激活函数是$sigmoid$函数）确定更新哪些信息，然后通过一个$tanh$层向细胞状态中加入一个新的候选值向量。用数学公式表示的话就是：$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$, $C_t^* = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$。
- 更新信息。
- 确定输出信息：分为两步，首先通过一个$sigmoid$层确定输出哪些信息，然后将这些信息经过处理之后输出。

4.3 创新点

利用神经网络预测用户行为，进行文件的预存取来提升 DFS 性能。

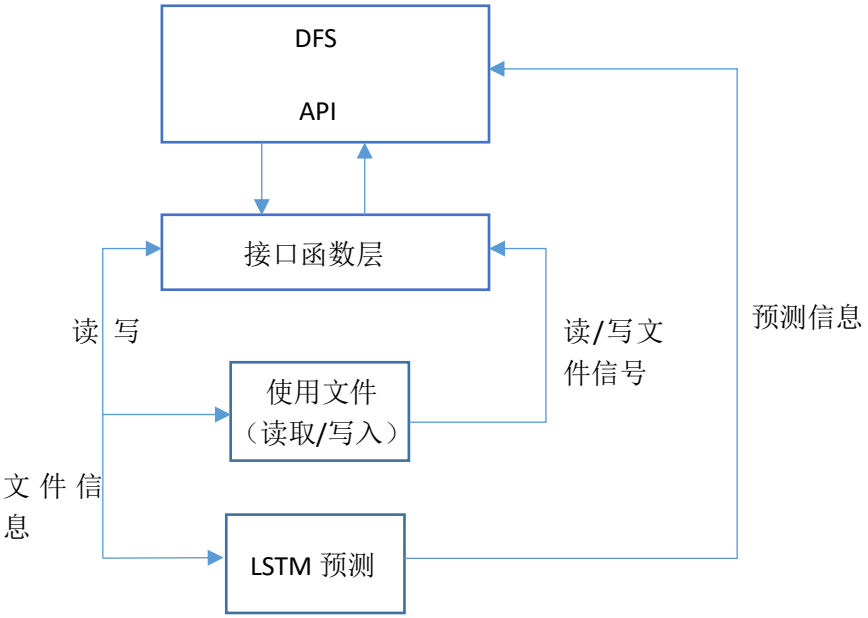
当然，通过文件预存取来提升性能的方法已经有很多论文提出过，但不同之处在于，我们采用了神经网络来预测，经过训练之后的神经网络的预测准确率应该是更为可观的。

4.4 总结

种种成功的预测案例表明，通过预测，确实很有可能提高 DFS 海量小文件存取的效率，而不是受限于硬件的瓶颈。而神经网络，主要是 LSTM，给我们提供了预测用户行为的手段，因此，我们认为这个课题的可行性是比较高的。

5 概要设计

5.1 总体设计



图八 总体设计框架

5.2 DFS 文件存取信息的获取

5.3 LSTM 的训练和预测

5.4 DFS 预取

6 参考文献

http://colah.github.io/posts/2015-08-Understanding-LSTMs/