学号：78066011

姓名：蔡佩津

MapReduce 模型 与 Hadoop

介绍

什么是 MapReduce？

MapReduce 是一种并行编程模型，用于编写分布式应用程序，在 Google 上以可靠、容错的方式在大型集群（数千个节点）上高效地处理大量数据（兆字节数据集）。MapReduce 程序运行在一个 Apache 开源框架 Hadoop 上。

MapReduce 是基于 Java 的分布式计算的处理技术和程序模型。MapReduce 算法包含两个重要任务，即 Map 和 Reduce。 Map 获取一组数据并将其转换为另一组数据，其中各个元素分解为元组（键/值）。其次，reduce 任务，它将地图的输出作为输入并将这些数据元组合并为较小的元组集。正如名称 MapReduce 的顺序所暗示的那样，reduce 任务始终在映射作业之后执行。

MapReduce 的主要优点在于，它易于在多个计算节点上扩展数据处理规模。在 MapReduce 模型下，数据处理原语称为映射器(mapper)和简化器(reducer)。有时将数据处理应用程序分解为映射器和约简器并非易事。但是，一旦我 们以 MapReduce 形式编写了一个应用程序，就可以将该应用程序扩展 为在集群中的数百台，数千台甚至数万台计算机上运行，这仅仅是配 置更改。这种简单的可伸缩性吸引了许多程序员使用 MapReduce 模型。

什么是 Hadoop?

Hadoop 是一个能够对大量数据进行分布式处理的软件框架，实现了 Google 的 MapReduce 编程模型和框架，能够把应用程序分割成许多的小的工作单元，并把这些单元放到任何集群节点上执行。在 MapReduce 中，一个准备提交执行的应用程序称为“作业（job）”，

而从一个作业划分出 得、运行于各个计算节点的工作单元称为“任务（task）”。此外，Hadoop 提供的分布式文件系统（HDFS）主要负责各个节点的数据存储，并实现了高吞吐率的数据读写。

1. 技术的产生背景

MapReduce的历史属于GFS，其主要用于满足Google对其数据处理需求快速增长的需求。另一方面，HDFS的目的是运行Hadoop MapReduce应用程序。这些是为具有不同需求的不同客户创建的。文件结构：GFS分为64MB块。由64位句柄标识的块，并被复制（默认三次）。它们进一步分为64 KB块，每个块都有一个校验和。在HDFS中，我们有128 MB的块。 NameNode将副本保存为两个文件：一个用于数据，一个用于校验和生成标记。

传统方法

在这种方法中，企业将拥有一台计算机来存储和处理大数据。对于数据库的存储，IBM等程序员会根据自己的选择来处理数据。

局限性(Limitation)

这种方法适用于那些处理少量数据的应用程序，这些数据可以由标准数据库服务器容纳，或者不超过处理数据的处理器的限制。但是，当涉及处理大量可伸缩数据时，通过单个数据库瓶颈处理此类数据是一项繁重的任务。

Google使用MapReduce的算法解决了这个问题。该算法将任务分为几部分，然后将它们分配给许多计算机，并从中收集结果，这些结果集成后便形成结果数据集。

2. 技术的基本原理

MapReduce编程范例使您可以在Apache Hadoop集群中的数百或数千个商用服务器上扩展非结构化数据。它具有两个主要组件或阶段，映射阶段和简化阶段。输入数据被馈送到映射器阶段以映射数据。

加载要在Map过程中进行的计算的程序称为Map函数，加载Reduce过程中进行的计算的程序称为Reduce函数。所以一个要运行MapReduce的程序员必须创建一个Map和Reduce函数的程序。

Map函数的功能是以键/值对的形式读取输入，然后以键/值对的形式生成输出。键/值对此映射函数的结果称为键/值中间对。然后，Reduce函数将读取Map函数结果的键/值中间对，并基于该键将其合并或分

组。换句话说，具有相同键的每个值都将合并到一个组中。Reduce函数还以键/值对的形式生成输出。

Hadoop 使用（Master/Slave）主从架构进行分布式储存和分布式计算。Master 负责分配和管理任务，Slave 负责实际执行任务。

在分布式存储和分布式计算方面，Hadoop 都是用从(Master/Slave)架构。在一个配置完整的集群上，想让 Hadoop 这头大象奔跑起来，需要在集群中运行一系列后台(daemon)程序。不同的后台程序扮演不用的角色，这些角色由 NameNode、DataNode、Secondary NameNode、JobTracker、TaskTracker 组成。其中 NameNode、Secondary NameNode、JobTracker 运行在 Master 节点上，而在每个 Slave 节点上，部署一个 DataNode 和 TaskTracker，以便 这个 Slave 服务器运行的数据处理程序能尽可能直接处理本机的数据。对 Master 节点需要特别说明的是，在小集群中，Secondary NameNode 可以属于某个从节点；在大型集群中，NameNode 和 JobTracker 被分别部署在两台服务器上。

基本算法（Basic Algorithm）

通常，MapReduce范例基于将计算机发送到数据所在的位置。

-MapReduce程序分三个阶段执行，即地图阶段，随机播放阶段和缩小阶段。

-Map阶段-Map或Mapper的工作是处理输入数据。通常，输入数据采用文件或目录的形式，并存储在Hadoop文件系统（HDFS）中。输入文件逐行传递到映射器功能。映射器处理数据并创建几个小数据块。

-Reduce阶段-此阶段是Shuffle阶段和Reduce阶段的组合。 Reducer的工作是处理来自映射器的数据。处理后，它将产生一组新的输出，这些输出将存储在HDFS中。

-在MapReduce作业期间，Hadoop将Map和Reduce任务发送到集群中的相应服务器。

-该框架管理所有数据传递的细节，例如发布任务，验证任务完成以及在节点之间的集群周围复制数据。

-大多数计算都在本地磁盘上有数据的节点上进行，从而减少了网络流量。

-完成给定任务后，集群将收集并减少数据以形成适当的结果，然后将其发送回Hadoop服务器。

逻辑视图(Logical view)

Map会在一个数据域中获取一对具有类型的数据，并返回不同域中的对列表：

Map(k1,v1)->list(k2,v2)

map函数与输入数据集中的每个部分并行应用，从而为每个调用生成一个成对列表。之后，MapReduce框架从所有列表中收集具有相同键的所有对并将它们分组在一起，为每个键创建一个组。

然后将reduce函数并行应用于每个组，这反过来又在同一域中产生值的集合：

Reduce(k2, list(v2))->list(v3)

每个Reduce调用通常会产生一个值v2或一个空返回，尽管允许一个调用返回多个值。所有调用的返回将收集为所需的结果列表。

这个MapReduce框架将（key，value）对列表转换为值列表。此行为不同于典型的函数编程map和reduce组合，后者接受一列任意值并返回一个单个值，该值组合了map返回的所有值。

输入和输出（Input and output）

MapReduce框架仅在<key，value>对上运行，也就是说，该框架将作业的输入视为一组<key，value>对，并生成一组<key，value>对作为作业的输出，可能是不同类型的。键和值类必须由框架可序列化（serializable），因此需要实现Writable接口。此外，key类必须实现WriteableComparable接口，以利于框架排序。

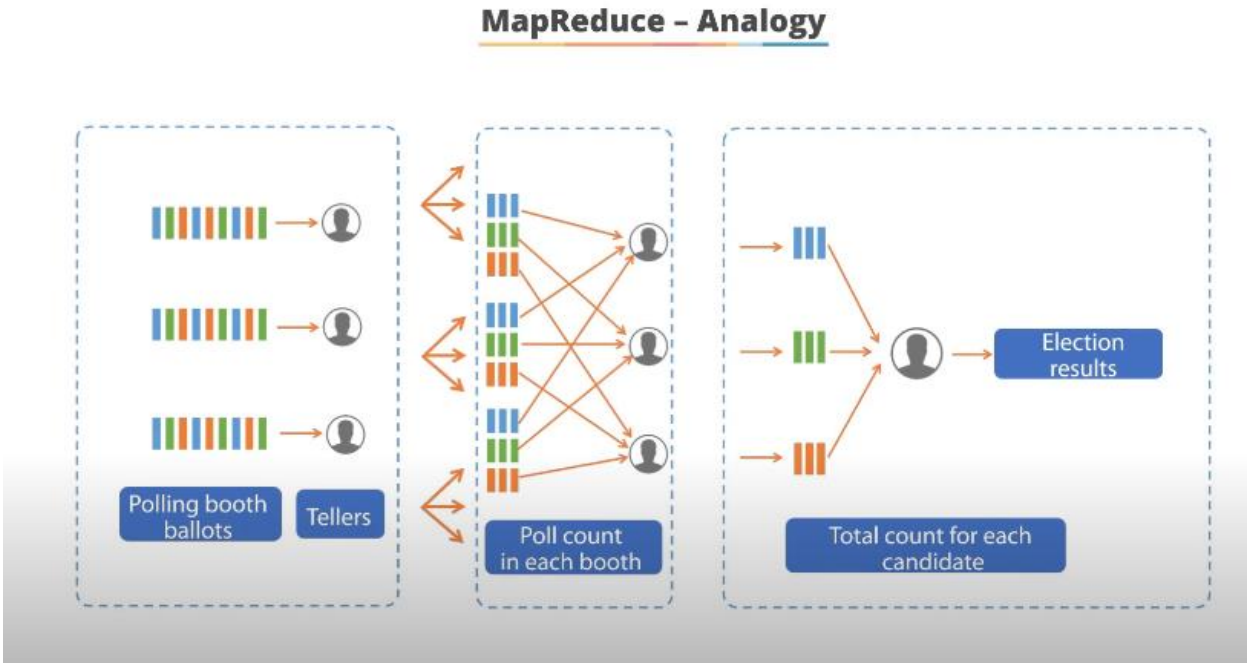
MapReduce作业的输入和输出类型：

(input)<k1,v1>->map<k2,v2>->combine-><k2,v2>->reduce-><k3,v3>(output)

假设我们将创建一个MapReduce程序来计算几个大文本文件中每个单词的单词数。在此程序中，可以如下定义Map函数和Reduce函数：

```
Function map(String key, String value):
//key: 文本文件的名称。
// value：文本文件的内容。
for each word W in value:
    emitIntermediate(W,"1");
function reduce(String key, Iterator values):
        int result = 0;
for each v in values:
    result+=ParseInt(v);
```

```
emit(AsString(result));
```



大选后算作类比(Counting after an election as an analogy)

第一步，柜员对每个投票站的选票进行计数，这称为输入拆分(input splitting)。

在第二步中，所有展位的柜员并行地计算选票，因为多个柜员正在从事一项工作，执行时间会更快，这称为映射方法(map method)。

在第3步中， 找到大会和议会的席位，并生成候选人的总数，这被称为简化方法。

因此，与单个计数器相比，可以映射并减少帮助以更快地执行作业。

3. 技术的特点

1. 可扩展性（Scalable）

Hadoop 是一个高度可扩展的平台。这主要是因为它具有跨大量服务器存储和分发大型数据集的能力。这些服务器价格便宜，可以并行运行。每增加一台服务器，就会增加更多的处理能力。

与无法扩展以处理海量数据的传统关系数据库管理系统（RDMS）相反，Hadoop MapReduce 编程使业务组织能够从大量节点中运行应用程序，这些节点可能涉及数千 TB 数据的使用。

2.安全与认证(Data Security)

安全性是任何应用程序的重要方面。如果任何非法人或组织可以访问组织中数 PB 的数据，则可能对您的业务交易和运营造成巨大伤害。

在这方面，MapReduce 与 HDFS 和 HBase 安全性一起使用，仅允许经过批准的用户对系统中存储的数据进行操作。

3.可用性和弹性(Availability and Flexibility)

当数据发送到整个网络中的单个节点时，同一组数据也将转发到组成网络的其他众多节点。因此，如果存在影响特定节点的任何故障，则始终可以在需要时仍可以访问其他副本。这总是确保数据的可用性。

Hadoop 提供的最大优势之一是其容错能力。 Hadoop MapReduce 能够快速识别发生的故障，然后应用快速，自动的恢复解决方案。在大数据处理方面，这使其成为游戏规则的改变者。

4.并行处理与容错(Parallel and Fault tolerance)

-Hadoop 框架允许用户快速编写和测试分布式系统。它是高效的，并且可以在计算机之间自动分配数据和工作，进而利用 CPU 内核的底层并行性。

-Hadoop 不依靠硬件来提供容错和高可用性（FTHA），而是 Hadoop 库本身已被设计为在应用程序层检测和处理故障。

-可以动态地在集群中添加或删除服务器，并且 Hadoop 可以继续运行而不会中断。

5.简单的编程模型(Simple programming)

Hadoop 的另一个重要优点是，除了开源以外，由于它基于 Java，因此在所有平台上都兼容。

应用场景：算法，高斯分析(Gaussian analysis)，Semantic Web 3.0，数据挖掘(Data mining)，搜索引擎操作(search engine operation)，排序，企业分析(enterprise analytics)

结论

当处理大型数据集时，Hadoop 的 MapReduce 编程允许以完全安全且经济高效的方式处理如此大量的数据。在处理大型数据集群方面，Hadoop 还胜过关系数据库管理系统。最终，许多企业已经实现了 Hadoop 所抱有的希望，而且随着非结构化数据的不断增长，它对企业的价值也必将增长。

基本术语

HDFS(Hadoop Distributed File System)：

HDFS Client：进行文件的分块与文件的发送读取。

Namespace image：记录每个文件的存在位置信息。

Edit log：记录每个文件的位置移动信息。

Namenode(Master)：管理着每个文件中各个块所在的数据节点的位置信息。

Secondary Namenode：更新并备份 Namenode。

Datanode(Slave)：记录着服务器内所储存的数据块的列表。

MapReduce：

Payload: 应用程序通常实现 Mapper 和 Reducer 接口以提供映射和 reduce 方法。这些构成了工作的核心。

Mapper: 将输入键/值对映射到一组中间键/值对。

Reducer: 减少一组共享一个较小值的键的中间值。

Reporter：MapReduce 应用程序用于报告进度，设置应用程序级别状态消息和更新计数器的工具。

OutputCollector: MapReduce 框架提供的功能的概括，用于收集 Mapper 或 Reducer 输出的数据

JobClient：用于把用户的作业任务生成 Job 的运行包，并存放到 HDFS 中。

JobinProgress：把 Job 运行包分解成 MapTask 和 ReduceTask 并存放于 TaskTracker 中。

JobTracker(Master)：进行调度管理 TaskTracker 执行任务。

TaskTracker(Slave)：执行分配下来的 Map 计算或 Reduce 计算任务

参考文献

https://www.guru99.com/introduction-to-mapreduce.html

https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.html

https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

https://blog.csdn.net/wuya814070935/article/details/78664674

https://en.wikipedia.org/wiki/MapReduce