**Parallel Data Processing with MapReduce: A Survey**

摘要

随着要分析的数据量快速增长，一种著名的并行数据处理工具 MapReduce 正从工业界和学术界获得巨大的动力。 虽然 MapReduce 用于许多需要大量数据分析的领域，但对其性能、每个节点的效率和简单抽象仍有争议。这项 调查旨在帮助数据库和开源社区理解 MapReduce 框架的各个技术方面。在本调查中，描述了 MapReduce 框架的特征，并讨论了其固有的优缺点。然后我们介绍了 ，最近文献中报道的优化策略。我们还讨论了在使用 ，MapReduce 进行并行数据分析时出现的开放问题和挑战。

1. 介绍

在这个数据爆炸的时代，并行处理对于及时处理海量数据至关重要。由谷歌推广的 MapReduce 是一种可扩展的容错数据处理工具，能够与许多低端计算节点并行处理大量数据。凭借其简单性、可扩展性和容错性，MapReduce 正在变得无处不在，并从工业界和学术界获得了巨大的动力。然而，MapReduce 对其性能和效率有固有的限制。因此，许多研究试图克服MapReduce 框架的局限性本次调查的目的是对 MapReduce 研究和相关工作的现状提供及时的评论，重点是当前旨在改进和增强MapReduce 框架的研究。概述了主要的方法，并根据它们的策略对它们进行了分类。

2. 体系结构

MapReduce 是一个编程模型，也是支持该模型的框 架。MapReduce 模型的主要思想是隐藏并行执行的细节，并允许用户只关注数据处理策略。MapReduce 模 型由两个基本函数组成:映射和缩减。MapReduce 的输 入是一个(key1，value1)对的列表，Map()应用于每个 对，以计算中间键值对(key2，value2)。中间键值对 然后在 keyequality 的基础上被分组在一起，即。 (key2，list(value2))。对于每个键 2，Reduce()处理所 有值的列表，然后生成零个或多个聚合结果。用户可 以定义 Map()和 Reduce()函数，只要他们希望 MapReduce 框架能够工作。MapReduce 利用谷歌文件系统(GFS)作为底层存储层 来读取输入和存储输出。GFS 是一个基于块的分布式文件系统，通过数据分区和复制支持容错。 Apache Hadoop 是 MapReduce 的开源 Java 实现。 用 Hadoop 进行解释，因为谷歌的 MapReduce 代 码不能供公众专有使用。其他实现(比如Erlang写的

DISCO)也有，但是没有 Hadoop 普及。像 MapReduce 一 样 ，Hadoop 由 两 层 组 成:一 个 称 为 Hadoop DFS(HDFS)的 数 据 存 储 层 和 一 个 称 为 Hadoop MapReduce 框架的数据处理层。HDFS 是一个由单个 主节点管理的块结构文件系统，就像谷歌的 GFS 一样。 Hadoop 中的每个处理作业被分解为与输入数据块一样 多的映射任务和一个或多个缩减任务。图 1 展示Hadoop 体系结构的概述。

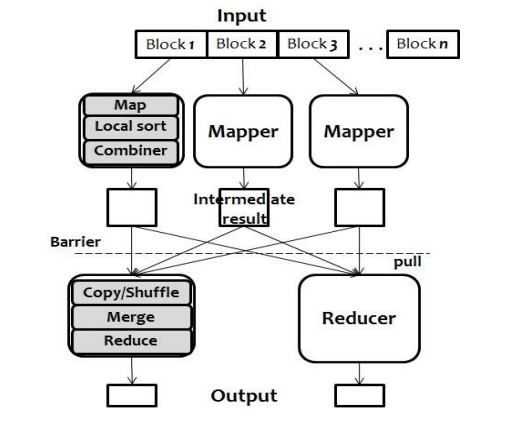
单个地图缩减作业分两个阶段执行:map和reduce阶段。 根据阶段给每个人分配 map或一个reduce任务。在开始映射任务之前，在分布 式文件系统上加载一个输入文件。在加载时，文件被 分割成多个大小相同的数据块，通常为 64MB，每个 数据块都是三重的，以保证容错。然后将每个数据块 分配给一个映射器，一个分配了映射任务的工作者， 映射器将映射()应用于数据块中的每个记录。然后，

映射器产生的中间输出在本地进行排序，以便对共享相同键值对进行分组。在本地排序之后，可以

选择应用 Combine()对分组的键值对执行预聚合，以 便将所有中间输出传输到缩减器所需的通信成本最小 化。然后，映射的输出被存储在映射器的本地磁盘中，并被分割成 R，其中 R 是作业中减少任务的数 量。这种划分基本上由散列函数来完成。，hash(key) mod R。 所有map任务完成后，map缩减计划程序会将缩减任 务分配给工作人员。中间结果被混洗，并通过 HTTPS 协 议分配给还原器。由于所有映射的输出已经被分区并存 储在本地磁盘中，每个缩减器通过简单地从映射器中拉

出映射输出的分区来执行混洗。基本上，映射输出的每 个记录通过一对一的洗牌策略只分配给一个缩减器。请 注意，该数据传输是通过减速器拉动中间结果来执行的。 缩减器读取中间结果并通过中间键合并它们，即。*key2， 以便将同一个键的所有值组合在一起。*这种分组是通过 外部合并排序完成的。然后，每个缩减器对它遇到的每 个键 2 的中间值应用 Reduce()。减压器的产量在 HDFS 储存并增加两倍。请注意，map任务的数量不取决于节点的数量，而是 取决于输入块的数量。每个块被分配给一个map任务。 但是，所有map任务不需要同时执行，减少任务也不需 要同时执行。例如，如果一个输入被分解成 400 个块， 并且一个集群中有 40 个映射器，则映射任务的数量为

400，并且映射任务通过 10 波任务运行来执行。MapReduce 框架基于运行时调度方案执行任务。这意 味着 MapReduce 在执行之前不会构建任何指定哪些任务 将在哪些节点上运行的执行计划。当数据库管理系统为 执行生成查询计划树时，MapReduce 中的执行计划完全 是在运行时确定的。通过运行时调度，MapReduce 通过 检测故障并将故障节点的任务重新分配给集群中的其他 健康节点来实现容错。完成任务的节点被分配了另一个输入块。该方案自然实现了负载平衡， 因为在下一波执行中，速度较快的节点将处理更多的 输入块，而速度较慢的节点处理较少的输入。此外MapReduce 调度程序利用了一种推测性的冗余执行。 分散节点上的任务在已经完成分配任务的其他空闲节 点上冗余地执行，尽管不能保证任务在新分配的节点 上比分散节点上更早结束。执行映射和缩减任务时， 其他任务之间没有通信。因此，在磁流变作业执行期 间，没有由同步引起的争用，也没有任务之间的通信成本。



截屏2020-12-20 下午9.04.30

3. 利弊

3.1 讨论

正如许多研究人员所建议的那样，商业数据库管理系 统采用了“一刀切”的策略，不适合解决极其大规模的数 据处理任务。人们一直需要为这些问题量身定制的专 用数据处理工具。虽然 MapReduce 被称 为数据中心计算中处理大数据的新方法，但与 DBMS 相比，它也被批评为并行数据处理中“重大倒 退”然而，业内许多 MapReduce 支持者认为 MapReduce 不是DBMS，这种苹果与橘子的比较是不公 平的。随着技术辩论的继续，ACM 最近在 2010 年 1 月版《CACM》中邀请双方[51，39]。DOLAP'10 中的小 组还讨论了 MapReduce 和关系数据库对于数据仓库的 利弊。

3.2 优势

MapReduce 是计算聚合的简单而有效的方法。因此， 它经常被比作数据库管理系统中的“先过滤后分组汇总” 查询处理。下面是 MapReduce 框架在数据处理方面的 主要优势。

**简单易用 MapReduce 模型简单但富于表现力。**

**灵活的 MapReduce 不依赖于任何数据模型和模式。**

**独立于存储 MapReduce 基本上独立于底层存储层。**

**容错 MapReduce 是高度容错的。**

**高可伸缩性使用 MapReduce 的最佳优势是高可伸缩**

**性。**

3.3 缺陷

尽管有许多优点，MapReduce 缺乏一些已被证明对 数据库管理系统中的数据分析至关重要的特性。在这 方面，MapReduce 通常被描述为提取-转换-加载(ETL) 工具[51]。与数据库管理系统相比，下面我们列举了 MapReduce 框架的缺陷。

**没有高级语言 MapReduce 本身不支持任何高级语言，**

**比如 DBMS 中的 SQL 和任何查询优化技术。**

**没有模式和索引 MapReduce 是无模式和无索引的。**

**单一固定数据流 MapReduce 通过简单的抽象提供了易 用性，但在固定数据流中。**

**低效率以容错和可扩展性为主要目标，MapReduce操 作并不总是针对输入/输出效率进行优化。**

**非常年轻的 MapReduce 从 2004 年开始被 Google 普及。**

**4. VARIANTS AND IMPROVEMENTS**

4.1 高级语言

Hive 是一个开源项目，旨在 Hadoop 之上提供数据 仓库解决方案，使用一种类似于 SQL 的查询语言 HiveQL 来支持即席查询。Hive 将 HiveQL 查询编译成 MR 作业的有向无环图(DAG)HiveQL 包括自己的类型 系统和数据定义语言(DDL)来管理数据完整性。它还 包含一个系统目录，包含模式信息和统计信息，非常 像 DBMS 引擎。Hive 目前只提供了一个简单的、基于 规则的优化器。 类似地，DryadLINQ被开发用来将程序的 LINQ 表达式翻译成微软的并行数据处理工具得律阿德 斯的分布式执行计划。

4.2 模式支持

如第 3.3 节所述，MapReduce 不提供任何模式支持。 因此，MapReduce 框架在读取输入时解析每个数据记 录，导致性能下降[15，51，11]。与此同时，姜等人 报告说，只有将记录转换为不可变数据对象的不可变 解码会严重导致性能下降，而不是记录解析。 虽然 MapReduce 本身不提供任何模式支持，但可以 使用谷歌的协议缓冲区、XML、JSON、Apache 的节俭 或其他格式来检查数据完整性。关于格式的一个 值得注意的事情是，它们是自描述格式，支持嵌套和 不规则的数据模型，而不是关系模型。使用这些格式 的一个缺点是，随着数据本身包含模式信息，数据大 小可能会增加。数据压缩被认为解决了数据大小问题

4.3 灵活的数据流

有很多算法很难直接映射到map和Reduce函数中。 例如，一些算法在处理过程中需要全局状态信息。循 环是一个典型的例子，它需要执行和终止的状态信息。 但是，MapReduce 在执行过程中不处理状态信息。因 此，MapReduce 迭代读取相同的数据，并在每次迭代 中在本地磁盘中实现中间结果，这需要大量的输入/输 出 和 不 必 要 的 计 算 。HaLoop[66]、Twister[42]和 Pregel[36]是在 MapReduce 中支持循环程序的系统示 例。 HaLoop 和 Twister 通过在迭代过程中识别并保持不 变的数据，避免了重复读取不必要的数据。类似地， 林等人提出了一种内映射组合技术，该技术在多个映 射调用之间将映射输出保存在内存缓冲区中，并在最 后一次迭代时发出聚合输出。此外，Twister 避免 了在迭代过程中重复实例化工人。以前实例化的工人 在 Twister 中用不同的输入在下一次迭代中被重用。 HaLoop 类似于 Twister，它还允许缓存每个阶段的输 入和输出，以在迭代过程中节省更多的输入/输出。普 通 Hadoop 还支持任务 JVM 重用，以避免为每个任务

启动新 JVM 的开销。Pregel 的主要目标是处理图 形数据。众所周知，图形数据处理通常需要大量迭代。 Pregel 实现了一个由批量同步并行(BSP)模型驱动的编 程模型。在这个模型中，每个节点都有自己的输入， 并且只将下一次迭代所需的一些消息传递给其他节点。 MapReduce 读取单个输入。然而，许多重要的关系 运算符是需要两个输入的二元运算符。映射-缩减-合 并通过简单地在缩减段之后添加第三个合并阶段来 解决关系运算符的支持问题。合并阶段将两个不 同磁共振作业的两个缩减输出合并为一个。 Clustera，得律阿德斯和 Niffele/PACT 允许比 MapReduce 更灵活的数据流。集群是 一个集群管理系统，旨在处理各种作业类型，包括 MRstyle 作 业。Clustera 中 的 作 业 调 度 器 处 理 MapReduce、工作流和 SQL 类型的作业，每个作业可以 连接起来形成一个 DAG 或一个用于复杂计算的管道。 得律阿德斯是分布式数据并行工具的一个显著例子， 它允许根据用户的意愿设计和执行数据流图。得律 阿德斯的数据流有一种由顶点和通道组成的 DAG 形式。 每个顶点代表一个程序，一个通道连接顶点。为了执行， 逻辑数据流图在运行时由作业调度器映射到物理资源上。 当一个顶点的所有输入都准备就绪时，它运行，并通过 数据流图中定义的通道将其结果输出到相邻顶点。通道 可以是文件、传输控制协议管道或共享内存。作业执行 由中央作业调度程序控制。冗余执行也被允许处理明显 非常慢的顶点，比如 MapReduce。得律阿德斯还允许定 义如何具体洗牌中间数据。 Nephele/PACT 是另一个并行执行引擎及其编程模型 。PACT 模型扩展了 MapReduce，以支持更灵 活的数据流。在该模型中，每个映射器可以有一个单独 的输入，用户可以通过更多不同的阶段指定其数据流， 包括映射和缩减。Nephele 将 PACT 程序转换成物理 DAG，然后跨节点执行 DAG。霞石中的执行是在运行时 调度的，就像 MapReduce 一样。

。

4.8 性能调整

大多数 MapReduce 程序都是为数据分析而编写的， 通常需要很长时间才能完成。因此，为 MapReduce 程 序提供自动优化功能是非常简单的。Babu 等人提出了 一种自动调谐方法，为给定的输入数据找到最佳的系 统参数[5]。它基于对采样数据的推测性预运行。 Jahani 等人提出了一种静态分析方法，称为 MANIMAL，用于单个 MapReduce 作业的自动优化[34]。在他们的 方法中，分析器在没有任何运行时信息的情况下，在 执行之前检查程序代码。根据分析过程中发现的规则，

它创建一个预先计算的 B+-树索引，并按列对输入数 据进行切片，供以后使用。此外，一些语义感知压缩 技术用于减少I/O。它的局限性是优化只考虑选择和投 影，这些主要在map功能中实现。

6. 讨论和挑战

MapReduce 正在变得无处不在，尽管它的效率和性能有争议。MapReduce 中使用的原则并无新

意。然而，MapReduce 表明，许多问题可以在模型中以前所未有的规模得到解决。由于频繁的检查点和带有推测性执行的运行时调度，MapReduce 显示出低效率。然而，这种方法对于在海量数据处理中实现高可扩展性和容错性是必要的。因此，如何提高效率，同时保证同等水平的可扩展性和容错性是一项重大挑战。效率问题有望通过两种方式克服:改进MapReduce 本身和利用新硬件。如何利用现代硬件的特性，在很多领域都没有得到回答。然而，芯片级多处理器和固态硬盘(SSD)等现代计算设备可以帮助大幅减少 MapReduce中的计算和 I/O。简单安装的 Hadoop 中 SSD 的使用被

简要检查，但不详细。多用户环境中的自调整和作业调度是另一个尚未很好解决的问题。集群

的规模不断扩大。一个4，000 节点的集群已经不足为奇了。如何在多用户环境中高效地管理如此规模的集群中的资源也是一个挑战。

7. 结论

讨论了 MapReduce 的优缺点，并对其改进进行了分类。MapReduce 简单，但为海量数据处理提供了良好的可扩展性和容错性。然而，MapReduce 甚至不太可能取代 DBMS 用于数据仓库。相反，我们期望 MapReduce用可扩展和灵活的并行处理来补充 DBMS，用于各种数据分析，例如科学数据处理。尽管如此，效率，尤其是map缩减的输入输出成本仍然需要被解决以获得成功的 影响。