(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 104270412 A
(43)申请公布日 2015.01.07

(21)申请号 201410455411.X

(22)申请日 2014.09.09

(66)本国优先权数据
201410287652.8 2014.06.24 CN

(71)申请人 南京邮电大学
地址 210046 江苏省南京市栖霞区亚东新城区文苑路9号

(72)发明人 孙知信 谢怡 宫婧

(74)专利代理机构 南京知识律师事务所 32207
代理人 汪旭东

(51)Int.Cl.
H04L 29/08(2006.01)
G06F 17/30(2006.01)

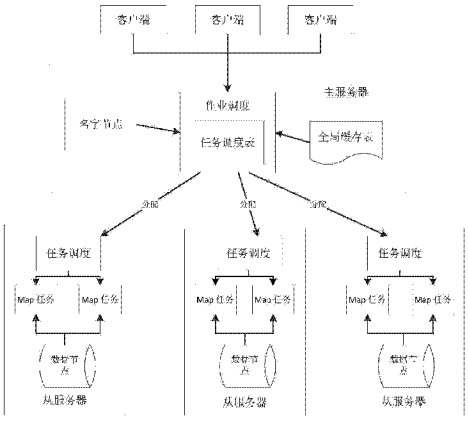
权利要求书1页 说明书5页 附图1页

(54)发明名称

一种基于Hadoop分布式文件系统的三级缓存方法

(57)摘要

本发明公开了一种基于Hadoop分布式文件系统的三级缓存方法，其包括三个大的步骤：步骤一、数据本地化处理的任务调度；步骤二、数据在本地内存的局部性访问；步骤三、本地内存数据的重复利用。本发明所提出的方法能够提高数据命中率，减少数据传输量，提升MapReduce的执行效率。



CN 104270412 A

1. 一种基于 Hadoop 分布式文件系统的三级缓存方法，采用 Apache Hadoop 实现，其方法如下：

步骤一、数据本地化处理的任务调度，又包含下列子步骤：

第 1 步、用户向 Jobtrack 提交作业请求，Jobtrack 获取 Job 所要读取的数据范围以及把作业分解成若干个 Map 任务和 Reduce 任务；

第 2 步、Jobtrack 根据每个 Map 任务所要读取的数据，通过访问 NameNode 的元数据来获取存放这些数据的 DataNode 位置；

第 3 步、空闲的 Tasktrack 节点定时向 Jobtrack 汇报自己的情况，Jobtrack 从这些空闲的 Tasktrack 节点中选择有目标数据的 DataNode，并将相应的 Map 任务分配到该节点中；

步骤二、数据在本地内存的局部性访问，又包含如下子步骤：

第 1 步、将服务器的内存空间分成大小相等的若干存储区域，每一块区域称之为页框；

第 2 步、每一页是最基本的内存分配，在每一页的底部预留字节来存放指向下一页的地址或者表示该数据块已结束，每个数据块在内存中表示为一串页的链表；

第 3 步、内存中维护一张数据块调入信息表，当内存中的数据块达到了存储空间的上限，新的数据块需要调入时，使用最近最久未使用置换算法来执行数据块的替换；另外，内存中还维护一张存储页面表的位示图；

步骤三、本地内存数据的重复利用：

第 1 步、在 Master 服务器中维护一张全局缓存信息管理表，负责记录每个 Slave 节点的缓存信息以及该节点是否有足够的 Slot 资源，每个 Slave 服务器定时向 Master 服务器发送信息，汇报自己的缓存信息和 Slot 资源信息；

第 2 步、当 Jobtrack 进行任务调度时，首先检查全局缓存信息表，如果发现缓存中有 Map 任务所需的数据，则优先分配该任务，倘若没有则遵循数据本地化处理的调度策略。

2. 根据权利要求 1 所述的一种基于 Hadoop 分布式文件系统的三级缓存方法，其步骤二中第 1 步中的叶框大小为 64k。

3. 根据权利要求 1 所述的一种基于 Hadoop 分布式文件系统的三级缓存方法，其步骤二中第 2 步中字节数为 4。

4. 根据权利要求 1 所述的一种基于 Hadoop 分布式文件系统的三级缓存方法，其步骤二中第 3 步最近最久未使用置换算法为选择最近一段时间内最长时间内没有被访问过的数据块进行置换。

5. 根据权利要求 1 所述的一种基于 Hadoop 分布式文件系统的三级缓存方法，其步骤二中第 3 步数据块调入信息表要记录的内容有数据块序号、数据块起始页框号、数据块访问时间。

6. 根据权利要求 1 所述的一种基于 Hadoop 分布式文件系统的三级缓存方法，其步骤二中第 3 步内存中还维护一张存储页面表的位示图，其方法为在内存中划分一块固定存储区域，每个内存单元的每个比特代表一个页面，如果该页面被分配，则对应的比特位置 1，否则置 0，以确定内存中是否存在空闲单元。

一种基于 Hadoop 分布式文件系统的三级缓存方法

技术领域

[0001] 本发明涉及数据存储领域，尤其涉及到一种基于 Hadoop 分布式文件系统的三级缓存方法。

背景技术

[0002] Apache Hadoop（一般简称为 Hadoop）是一个开源的分布式数据处理平台，它主要包括分布式文件处理系统 (Hadoop Distributed File System，HDFS) 和 MapReduce 计算模型。

[0003] Apache Hadoop 是一款支持数据密集型分布式应用并以 Apache2.0 许可协议发布的开源软件框架。它支持在商品硬件构建的大型集群上运行的应用程序。Hadoop 是根据 Google 公司发表的 MapReduce 和 Google 档案系统的论文自行实作而成。

[0004] Hadoop 框架透明地为应用提供可靠性和数据移动。它实现了名为 MapReduce 的编程范式：应用程序被分割成许多小部分，而每个部分都能在集群中的任意节点上执行或重新执行。此外，Hadoop 还提供了分布式文件系统，用以存储所有计算节点的数据，这为整个集群带来了非常高的带宽。MapReduce 和分布式文件系统的设计，使得整个框架能够自动处理节点故障。它使应用程序与成千上万的独立计算的电脑和 PB 级的数据。现在普遍认为整个 Apache Hadoop “平台”包括 Hadoop 内核、MapReduce、Hadoop 分布式文件系统 (HDFS) 以及一些相关项目，有 Apache Hive 和 Apache HBase 等等。

[0005] HDFS 能很好地满足大规模数据的存储需求，然而，它在处理实时数据读取方面存在许多不足。由于执行 MapReduce 任务的过程中涉及到大量的数据读取，会对网络传输和 I/O(Input/Output) 带宽造成巨大压力，所以要在 HDFS 的基础上设置缓存系统，减少数据传输量，以提高 MapReduce 的执行效率。

[0006] MapReduce 将数据计算过程分为两个阶段：Map 和 Reduce，对应于两个处理函数 mapper 和 reducer。在 Map 阶段，原始数据被输入 mapper 进行过滤和转换，获得的中间数据结果作为 reducer 的输入，得到最后的处理结果。在整个 MapReduce 的处理过程中，从 HDFS 中读取原始数据所花费的时间最长，因此，要想提高 MapReduce 的执行效率，需要从原始数据的读取入手。通过设置相应的缓存机制，提高数据命中率，使 Map 阶段原始数据的读取时间降低。

[0007] Memcached 和 RAMCloud（内存云）是两个典型的内存级缓存系。Memcached 是在基于磁盘存储的海量数据中为后端提供相对集中的、非协作的、对用户非透明的数据缓存服务。RAMCloud 则是使用分布式共享内存替代磁盘完成数据存储和管理，并提出了缓存数据基于多磁盘分散备份，并行快速修复的思想。

[0008] 两者本质上都还是面向磁盘数据相对集中存放、计算资源与存储资源分离的架构，因此难以直接适用于 MapReduce 平台；同时由于支持的应用类型差异，二者均未考虑由于 MapReduce 应用数据本地化处理的特征。

[0009] 基于时间局部性的缓存替换策略 LAC(Locality-Aware Cooperative Caching，位

置感知协作缓存）通过同一数据块的两次访问之间所间隔的其他数据块的访问次数来量化数据访问的时间局部性。将数据块的传输代价、数据块的大小以及数据块的最近访问时间等因素构建缓存替换代价模型。

[0010] 这些机制均面向传统数据中心平台架构，然而在 Map/Reduce 平台计算资源与存储资源的紧耦合部署和数据本地化处理的特征，使得基于数据块级的数据访问特征统计结果受到计算资源分配策略及实时负载的干扰，难以完整真实的反映数据访问特征。

[0011] 针对 MapReduce 任务执行过程中需要读取大量的数据，对网络传输和 I/O 带宽造成巨大压力，现有技术中上没有很好的解决方法。

发明内容

[0012] 为解决上述技术问题，本发明采用了如下技术方案：

[0013] 一种基于 Hadoop 分布式文件系统的三级缓存方法，采用 Apache Hadoop 实现，其方法如下：

[0014] 步骤一、数据本地化处理的任务调度，又包含下列子步骤：

[0015] 第 1 步、用户向 Jobtrack 提交作业请求，Jobtrack 获取 Job 所要读取的数据范围以及把作业分解成若干个 Map 任务和 Reduce 任务；

[0016] 第 2 步、Jobtrack 根据每个 Map 任务所要读取的数据，通过访问 NameNode 的元数据来获取存放这些数据的 DataNode 位置；

[0017] 第 3 步、空闲的 Tasktrack 节点定时向 Jobtrack 汇报自己的情况，Jobtrack 从这些空闲的 Tasktrack 节点中选择有目标数据的 DataNode，并将相应的 Map 任务分配到该节点中；

[0018] 步骤二、数据在本地内存的局部性访问，又包含如下子步骤：

[0019] 第 1 步、将服务器的内存空间分成大小相等的若干存储区域，每一块区域称之为页框；

[0020] 第 2 步、每一页是最基本的内存分配。在每一页的底部预留字节来存放指向下一页的地址或者表示该数据块已结束。每个数据块在内存中表示为一串页的链表。

[0021] 第 3 步、内存中维护一张数据块调入信息表，当内存中的数据块达到了存储空间的上限，新的数据块需要调入时，使用最近最久未使用置换算法来执行数据块的替换；另外，内存中还维护一张存储页面表的位示图。

[0022] 步骤三、本地内存数据的重复利用

[0023] 第 1 步、在 Master 服务器中维护一张全局缓存信息管理表，负责记录每个 Slave 节点的缓存信息以及该节点是否有足够的 Slot 资源。每个 Slave 服务器定时向 Master 服务器发送信息，汇报自己的缓存信息和 Slot 资源信息。

[0024] 第 2 步、当 Jobtrack 进行任务调度时，首先检查全局缓存信息表，如果发现缓存中有 Map 任务所需的数据，则优先分配该任务。倘若没有则遵循数据本地化处理的调度策略。

[0025] 步骤二中第 1 步中的叶框大小为 64k。

[0026] 步骤二中第 2 步中字节数为 4。

[0027] 步骤二中第 3 步最近最久未使用置换算法为选择最近一段时间内最长时间内没有被访问过的数据块进行置换。

[0028] 步骤二中第 3 步数据块调入信息表要记录的内容有数据块序号、数据块起始页框号、数据块访问时间。

[0029] 步骤二中第 3 步内存中还维护一张存储页面表的位示图，其方法为在内存中划分一块固定存储区域，每个内存单元的每个比特代表一个页面，如果该页面被分配，则对应的比特位置 1，否则置 0，以确定内存中是否存在空闲单元。

[0030] 本发明所提出的基于 HDFS 的三级缓存机制能够提高数据命中率，减少数据传输量，提升 MapReduce 的执行效率。

附图说明

[0031] 图 1 基于 HDFS 的三级缓存架构图。

具体实施方式

[0032] 本发明所提出的三级缓存机制是运行于 Hadoop 平台的分布式文件系统 HDFS，包括三个层次，如图 1 所示 ：1、数据本地化处理的任务调度 ；2、数据在本地内存的局部性访问 ；3、本地内存数据的重复利用。

[0033] 一般情况下，HDFS 包含一个 NameNode 部署在主服务器上大量 DataNode 部署在从服务器上 NameNode 负责管理用户文件的元数据信息。元数据由三部分组成，分别是文件系统目录树信息、文件和文件所拆分的数据块的对应关系、数据块在 DataNode 上的分布位置信息。存储在 HDFS 的文件被拆成同样大小的数据块（通常是 64MB），这些数据块将会复制存储到多个数据存储服务器中。每个数据存储服务器在本地磁盘上用 Linux 文件系统保存这些块，并且对数据块进行读写。

[0034] MapReduce 计算模式是一种标准的函数式编程模式。客户端用户通过编程实现对文件的操作。每个用户程序可以看做是一个作业，而作业将会由 Jobtrack（作业调度）分解成若干个 Map 和 Reduce 任务。Map 任务从 HDFS 中读数据，Reduce 任务处理好的数据写入 HDFS，所以本发明所提出的三级缓存系统主要服务于 Map 任务，保证 Map 能在最短的时间内找到想要的数据。

[0035] 步骤一 ：数据本地化处理的任务调度

[0036] 由于 HDFS 是一个 Master/Slave 结构（主从结构），因而可以将 NameNode（名字节点）和 Jobtrack 部署在 Master 服务器（主服务器）中，DataNode（数据节点）和 Tasktrack（任务调度）部署在 Slave 服务器从服务器中。假如 Map 任务所要处理的数据恰好保存在本地服务器上，那么 Map 任务可以直接从本地磁盘中读取数据，减少数据在网络中传输时间。当 Jobtrack 进行任务调度时，优先将 Map 任务分配给包含该任务所要处理数据块的 DataNode 上。为了实现这一目的，split 分片大小与数据块大小一致，因此在 InputSplit 元数据信息中，host 列表上只包含一个节点，能完全实现数据本地化处理。

[0037] 以数据本地化处理为导向的任务调度过程如下 ：

[0038] 1、用户编写 MapReduce 程序创建新的 JobClient 实例，向 Jobtrack 提交作业请求。Jobtrack 收到一个 Job Client 请求，并做应答。然后获取 Job 所要读取的数据范围以及把作业分解成若干个 Map 任务和 Reduce 任务，每个 Map 任务处理相应的部分数据，为一个 split 大小。

[0039] 2、Jobtrack 根据每个 Map 任务所要读取的数据，通过访问 NameNode 的元数据来获取存放这些数据的 DataNode 位置，包括备份的数据节点位置。

[0040] 3、空闲的 Tasktrack 节点定时向 Jobtrack 汇报自己的情况，Jobtrack 从这些空闲的 Tasktrack 节点中选择有目标数据的 DataNode，并将相应的 Map 任务分配到该节点中。

[0041] 步骤二：数据在本地内存的局部性访问

[0042] 显然，磁盘的读取速率远远跟不上 CPU 的处理速率，所以设置内存缓冲区来平衡两者的差距。将磁盘内的部分数据预先读取到内存中，当 CPU 遇到读取数据的指令时，直接从内存中获得相应的数据。这样可以减少 Map 任务从本地磁盘读取数据的时间，本发明所提出的内存级数据缓存调度是以局部性访问为基础。虽然两个 Map 任务之间是独立的，但是 Map 任务是从用户程序中分解出来的，所以前后之间满足局部性访问原理，可以将数据预先调入内存或者上次被读取的数据不要马上调出内存，在下一个 Map 任务到来之时，便可以直接读取数据。

[0043] 以局部性访问为基础的内存级数据缓存调度过程如下：

[0044] 1、将服务器的内存空间分成大小相等的若干存储区域，每一块区域称之为页框，页框大小为 64K，由于 HDFS 中每个数据块的大小为 64MB，所以一个数据块所需的页框数为 1024 个。页框在内存中从 0 开始连续编号。

[0045] 2、每一页是最基本的内存分配。在每一页的底部预留 4 个字节来存放指向下一页的地址或者表示该数据块已结束。每个数据块在内存中表示为一串页的链表。

[0046] 3、为了方便从内存中找到所需的数据块，从服务器的内存中维护一张数据块调入信息表，显示哪些数据块调入内存，在内存中的存储位置以及什么时候调入内存，因此该信息表中要记录的选项有：数据块序号、数据块起始页框号、数据块访问时间。当内存中的数据块达到了存储空间的上限，新的数据块需要调入时，使用最近最久未使用置换算法 (Least Recently Used，LRU) 来执行数据块的替换，即选择最近一段时间内最长时间内没有被访问过的数据块进行置换，因为访问局部性原理认为在过去一段时间内不曾被访问过的数据块，在最近的将来也不会被访问，所以数据块信息表中需要记录数据块的调入时间，是为执行 LRU 算法准备的另外，内存中还维护一张存储页面表的位示图，在内存中划分一块固定存储区域，每个内存单元的每个比特代表一个页面，如果该页面被分配，则对应的比特位置 1，否则置 0，以确定内存中是否存在空闲单元，位示图的作用是为了指出内存中各页面是否已被分配出去，以及为分配页面的总数。前面也提到了，为了方便内存的管理，将内存空间分成大小相同的若干页框，数据块调入内存中，必然要分配部分页来存储，那么为了管理页框的分配，所以用位示图来表示每一页的分配情况，分配了置“1”，没分配置“0”。下一次要调入数据块时，首先要查看位示图，看是否有足够的页框供数据块调入，假如不够则要执行 LRU 算法，把之前的数据块调出内存，空出页框，供后面的数据块调入。

[0047] 4、假设内存空间大小为 8GB，预留 2GB 的空间作为其他用途，那么真正可以用来做内存缓存的空间大小为 6GB。HDFS 每个数据块的大小为 64MB，所以内存缓存中一次性能调入数据块有 96 个数据块，所以数据块调入信息表不会很大。

[0048] 步骤三：本地内存数据的重复利用

[0049] Map 任务执行结束后，它所读取的数据会保留在内存中。当下一个 Map 任务分配到该 DataNode 时，为了避免内存替换所花费的不必要时间，Jobtrack 在下一轮分配任务时应

优先考虑节点内存中的数据与Map任务所需处理数据的匹配情况。所以本发明提出以内存数据重复利用为驱动的任务调度。

[0050] 以内存数据重复利用为驱动的任务调度过程：

[0051] 1、在Master服务器中维护一张全局缓存信息管理表，负责记录每个Slave节点的缓存信息以及该节点是否有足够的Slot资源。每个Slave服务器定时向Master服务器发送信息，汇报自己的缓存信息和Slot资源信息，TaskTracker使用slot表示在本节点上等量划分的资源量。slot代表计算资源(CPU、内存等)。一个Map Task获取到一个slot后才有机会运行。

[0052] 2、当Jobtrack进行任务调度时，首先检查全局缓存信息表，如果发现缓存中有Map任务所需的数据，则优先分配该任务。倘若没有则遵循数据本地化处理的调度策略。

[0053] 所以，根据本发明所提出的基于HDFS的三级缓存机制，Jobtrack在任务调度和数据加载时遵循如下过程：Jobtrack分配Map任务时，先检查全局缓存信息表，优先把任务分配给内存中调入目标数据块的节点；然后根据数据本地化处理的策略，将剩余的Map任务分配到保存目标数据块的节点中，执行Map任务时要把所需读取的数据块加载到内存中，内存替换遵循以局部性访问为基础的内存级数据缓存调度。

[0054] 本发明基于HFDS的三级缓存机制，包含以数据本地化处理为导向的任务调度、以局部性访问为基础的内存级数据缓存调度、以内存数据重复利用为驱动的任务调度。以局部性访问为基础的内存级数据缓存调度过程，数据块调入内存中是按页存储，每个数据块在内存中表示为一串页的链表，易于内存空间的回收利用。通过数据块调入信息表进行内存替换。

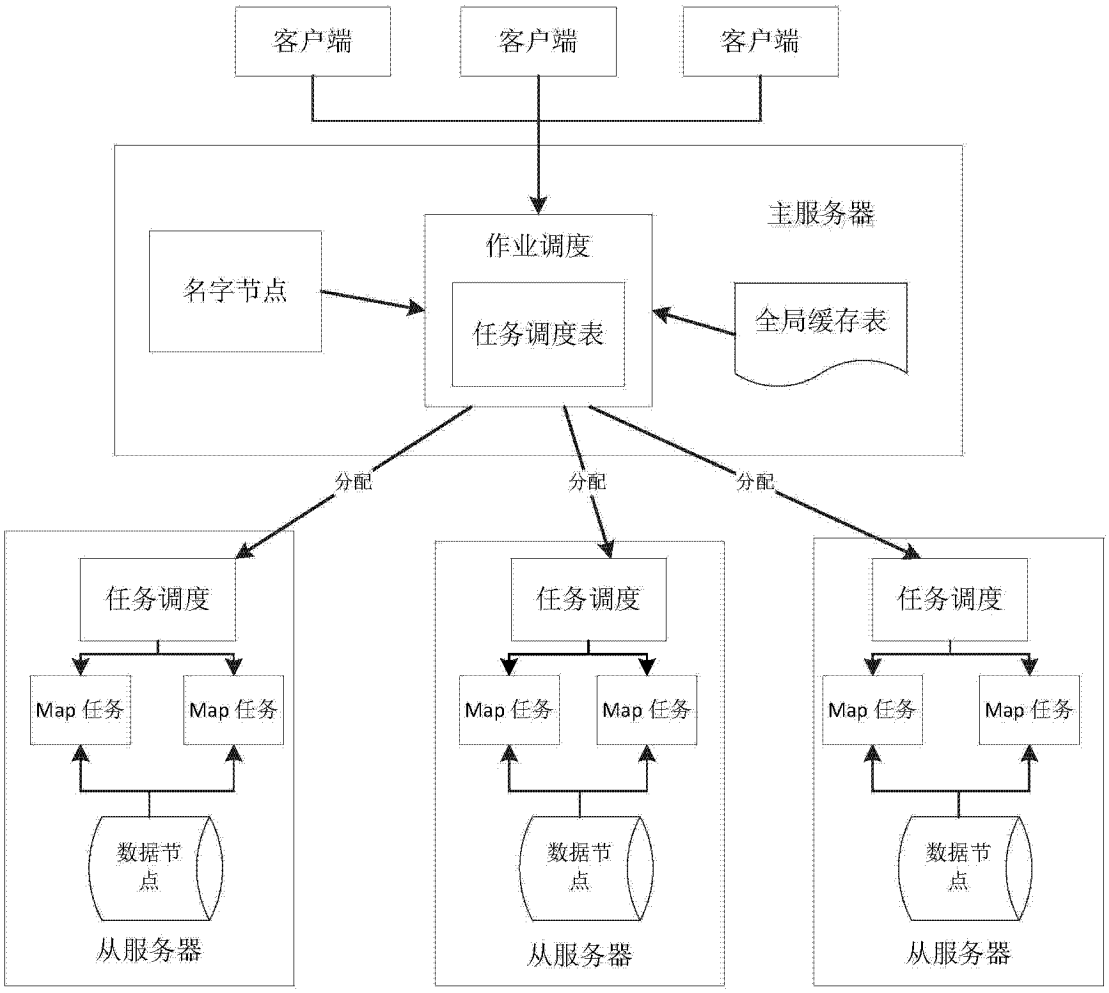

客户端
客户端
客户端
主服务器
作业调度
名字节点
任务调度表
全局缓存表
分配
分配
分配
任务调度
任务调度
任务调度
Map 任务
Map 任务
Map 任务
Map 任务
Map 任务
Map 任务
数据节点
数据节点
数据节点
从服务器
从服务器
从服务器

图 1