一种数据密集型应用的数据副本管理策略①

王 雷, 陶 伟

(中国科学技术大学 自动化系, 合肥 230027)

摘 要: 基于工作流的数据密集型应用是云计算环境下的一种常见应用. 当需要处理分布式存储在多个数据中心的数据时, 如何高效地获取数据, 是直接关系到流程执行效率和服务质量(QoS)的重要问题. 对数据密集型应用进行建模, 通过对数据节点负载匹配的测量, 设计了一种基于域自治的数据副本管理策略. 仿真实验表明, 该策略能较好地解决多数据中心协同计算时的数据获取效率问题.

关键词: 云计算; 数据密集; 负载匹配; 域自治; 副本管理

A Duplication Strategy for Data-Intensive Application

WANG Lei, TAO Wei

(Department of Automation, University of Science and Technology of China, Heifei 230027, China)

Abstract: Data-intensive computing application based on workflow accounts for a large share in cloud computing. When processing data being stored in more than one data center. How to get data efficiently plays a big part in improving quality of service(QoS) and process execution efficiency. In this paper, a model is presented to describe the data intensive application. By measuring matched load of data nodes, a domain-based duplication strategy is also declared out. At last,the simulation results show that this strategy can improve the data acquisition efficiency markedly.

Key words: cloud computing; data intensive; load matching; domain autonomy; replica management

1 引言

近年来随着通信技术和互联网技术的进步, 用户的网络行为模式发生了很大的变化, 由 Web 服务组合而成的 Web 应用已经成为网络的主流应用. 与此同时, 由于信息采集和传输技术的不断发展, 用户越来越多地开始传输和使用影像资料等消耗大量存储和带宽的信息数据. 因此, 通过互联网提供基于海量数据(TB 甚至是 PB 级)的服务已成为信息社会发展的一个重要趋势. 此时, 无论是互联网企业还是个人用户, 都比以往更加关注服务质量(QoS)问题.

互联网服务的质量很大程度上取决于在高并发请求下提供数据的快速性、准确性、时效性和全面性. 一方面, 每个服务背后都可能都会面对海量的数据, 其中既包括输入数据, 也包括中间数据和结果数据; 另一方面, 常见的互联网应用通常由若干个服务组合而成, 也就是一个可以用工作流描述的业务系统, 其流程可以是若干服务的简单串联或并联, 也可以是包含分支聚合结构的复杂逻辑组合, 这些都为服务质量的保证带来了很大困难. 在这种背景下, 卡内基-梅隆大学计算机学院院长 Bryant 提出了”Data-intensive scalable computing”[1]的研究计划, 从学术的角度阐述了一个新的研究领域——以海量数据处理为中心的数据密集型计算.

云计算概念的提出[2], 使得数据密集型计算技术的研究进入了一个新的阶段. 云计算强调系统提供服务的方式, 当服务与大量数据交互时, 数据管理及其副本策略成为云环境下计算任务能否高效完成的关键因素.

目前, 已经出现了一些云计算环境下的数据存储管理系统, 例如 Amazon S3[3], Google File System[4]和

① 基金项目:中央高校基本科研业务费专项资金(WK2100100012)
收稿时间:2012-04-25;收到修改稿时间:2012-06-05

Hadoop[5], 它们均对用户隐藏了用于存储应用数据的基础设施, 仅提供了相关的应用接口. 针对大多数用户而言, Amazon S3 就是短期或长期的备份设备, 用户可以不断地把数据放到里面, 而不用担心数据的完整性及存储空间的耗尽问题, 但 S3 并未关注数据访问速度; Google File System 主要针对 Web 搜索应用; Hadoop 则是一个更为通用的分布式文件系统.

包含业务逻辑的数据密集型应用需要在整个工作流程中处理很多份数据, 不同数据之间以业务流程中的服务为纽带而相互联系和相互依赖. 其中, 数据的存储布局策略直接影响服务获取数据的速度, 从而影响到整个业务流程的执行效率. 当前, 已有很多数据依赖性的相关研究. BitDew[6]通过由用户来定义数据属性的方式来体现数据的相关性. 但是在数据密集型计算环境下, 由于业务需求是不断变化的, 数据的规模也很大, 让用户更改数据属性并不现实. 文献[7]给出了一种基于聚类矩阵的数据布局策略, 用于多数据中心环境下数据布局方案的求解与优化, 该策略能减少云计算环境下流程应用执行过程中的跨数据中心的传输次数. 但该策略并未考虑网络带宽及数据中心存储设备的异构问题, 因此并不总是能很好的提高执行效率. 此外, 用户业务请求对数据的需求是实时变化的, 该策略在对处理数据中心海量的数据进行聚类分析, 计算量比较大, 具有一定的时滞性.

综上所述, 当前对云计算环境下数据密集型应用的数据存储布局问题的研究都具有一定的局限性, 往往只关注不同数据中心的传输次数, 而忽略了网络带宽及存储设备的异构性. 本文针对云环境下基于工作流的数据密集型应用的若干特点, 考虑存储数据的传输带宽、存储设备的容量和 I/O 速度以及数据的冗余部署等因素, 设计了一种域自治策略, 动态调整数据副本的存储位置, 从而提高系统整体性能.

2　问题描述与求解方法

2.1 基于工作流的数据密集型应用的模型

典型的数据密集型计算应用通常由若干个服务组成, 可以用工作流来描述. 应用流程在执行过程中调用服务, 服务获取源数据并产生中间数据. 当数据量很大时, 数据获取时间是最重要的性能指标. 图 1 是一个典型的数据密集型应用流程. 图中, 服务表示为 $S_i(i=1,\cdots n)$, 数据表示为 $d_j(j=1,\cdots m)$, 背景无斑点的数据表示源数据, 有斑点的数据表示服务生成的中间和结果数据. 将服务 S_i 读写数据的时间记为 t_{di}, 计算的时间记为 t_{ci}, 则其完成一次调用的总时间消耗 t_i 可表示为:

$$t_i = t_{ci} + t_{di} \tag{1}$$

因此, 数据密集型应用的性能指标可定义为总执行时间为 T:

$$\begin{aligned} T &= \sum t_i = \sum (t_{ci} + t_{di}) \\ &= \sum t_{ci} + \sum t_{di} \end{aligned} \tag{2}$$

为简化问题, 设每个服务的计算时间相同, 为常数 m, 则:

$$T = \sum t_{ci} + \sum t_{di} = nm + \sum t_{di} \tag{3}$$

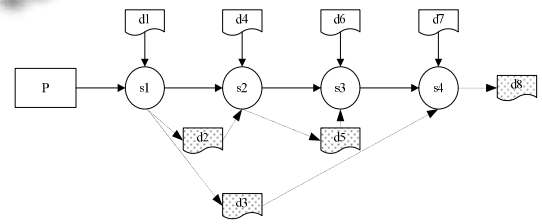


图 1　典型的数据密集型计算应用流程

2.2 负载匹配的测量;

云计算环境下, 服务节点和数据节点在逻辑上是分离的. 在不考虑服务迁移的情况下, 云环境下的数据密集型计算应用的性能指标同2.1, 即取决于从数据节点获取数据的效率. 诸如 Hadoop 等云计算存储系统都会在不同数据节点上维护多个工作数据副本, 以保证数据的可靠性并获得良好的性能. 常用的算法中, 在业务执行过程中获取数据时, 往往简单地选择同一机架或者距离最近的机架上存储的数据. 当系统业务请求量较低时, 这种简单的策略能取得很好的效果, 获得理想的 QoS, 但当遇到大量的特定服务请求时, 数据需求往往也比较集中, 上述策略就不一定能取得很好的效果. 本文参考文献[8]、[9], 将 2.1 中的性能指标转化为基于向量匹配的 QoS 测量, 并基于此提出负载匹配的概念.

在数据密集型应用中, 为获取数据的存储主要考虑的因素有传输带宽, 存储设备的I/O速度, 存储容量的使用率等因素. 定义服务执行节点和数据存储节点之间的最大网络带宽为 V, 已用带宽为 V_{used}, 存储设备的总容量为 C, 已使用容量为 C_{used}, 源数据大小为 D_i, 生成数据大小为 D_o, 存储设备的最大输出(读)速率为 P_o, 最大输入(写)速率为 P_i, 实时输出速率为

P_{oused}, 实时输入速率为 P_{iused}, 则总的资源向量为:

$$W = (V, C, P_o, P_i) \quad (4)$$

可用资源向量为:

$$W_a=(V-V_{used}, C-C_{used}, P_o-P_{used}, P_i-P_{iused}) \quad (5)$$

其中, $C_{used}=D_i+D_o$, 系统总体性能指标中的Σt_{di} 则对应为 $D_i/P_{oused} + D_o/P_{iused}$.

为了系统的负载均衡性, 达到充分发挥系统性能, 定义系统的负载率匹配度:

$$\Phi = \alpha \frac{W \cdot W_a}{|W|^2} + (1-\alpha)\frac{1}{f(W,W_a)} \quad (6)$$

其中, $0<\alpha<1$ 为匹配因子, f(W, W_a)表示 W, W_a 之间的夹角.

以下通过对负载匹配度的测量, 提出一种数据副本自适应管理策略.

2.3 基于域自治的数据副本管理算法

在面向互联网领域的数据密集型应用中, 数据存储的负载特性通常有如下特点[5]:

1) 读操作由大量的顺序读和少量的随机读组成;

2) 写操作绝大部分是在文件尾部的追加写, 对一个文件的随机写的情况几乎不会出现;

3) 数据一旦写完, 文件一般就是只读, 这种一次写入多次读取的特性对数据的一致性要就较弱;

4) 数据的访问频率通常在不断变化, 但具有一定的规律, 通常特定数据只是在一定区域、一定时间内被较多的访问.

一般情况下, 数据中心存储数据时都会采用副本策略, 在跨地域的数据分中心保存多个数据副本缓存, 以提高系统的性能和可靠性. 副本存储的布局对系统的性能影响十分明显, 如前所述, 目前有研究采用预定义[6]的方法实现数据布局, 也有采用聚类分析的方法[7], 对数据进行分析, 实现数据存储布局的优化. 但在云计算环境下, 面对不断增加的海量数据, 而且对数据的访问需求随用户需求不断变化, 上述研究方法很难达到较好的实时性. Hadoop 在写数据时, 随机选择存放位置, 并默认创建 3 个副本, 副本数据存放位置以后不会变化; 在读数据时, 根据数据节点到读取端的距离选择节点. 基于互联网中的请求特性, 这种策略下, 数据请求往往会集中在某一节点上, 导致获取数据的效率低下. 如上所述, 在面向互联网领域的数据密集型应用的数据存储中, 对数据文件的操作主要是读. 基于此, 本文提出了基于域的副本管理方法, 即, 利用 treap 树对负载敏感[10]的特性, 将每个数据与其副本组成一个 treap 树内的域, 域内副本根据服务的数据请求情况动态调整, 自动优化以适应需求, 存储数据的数据结构格式如下:

```
Struct FileBlock {
    Bool    SourceProperty = true;// true 代表
    // 文件是元数据,  false 代表文件是副本数据
    int     NoofCopy;       // 副本数量
    int     LoadMatching;   // 负载匹配度, 根据
                            // 式(6)计算
    File    FileKey;        // 文件标识
    FileBlock   * ParentFile; // 父节点
    FileBlock   *LeftChildFile;//左子节点
    FileBlock   *RightChildFile; //右子节点
}
```

SourceProperty 属性标识了数据是否是元数据, 元数据起到索引数据的作用, 不可移动、删除. 如图 2 所示, 当服务获取数据时, 所找到的第一个数据就是元数据, 然后在所有副本中实时计算负载率匹配度, 利用 treap 树的翻转操作, 保证高层次节点的负载匹配度较高, 方便选取最优的节点获取数据.

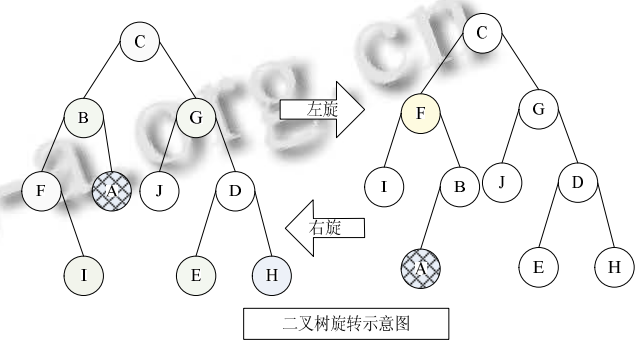


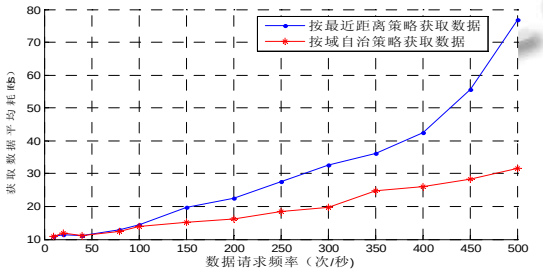
图 2 treap 树旋转示意图

我们考虑通过计算负载率匹配度, 并利用 treap 树特性调整二叉树的过程. 在保证副本数量在一定范围的前提下, 如果底层节点的负载率匹配度过低, 则依照 treap 树的删除操作予以删除. 如果高层次节点的负载率匹配度依然不满足要求, 由于该节点已经是当前最优, 则符合要求的节点肯定在其临近节点中, 在高层次节点所在的数据中心和其相邻的数据中心各随机选择一个节点, 然后在两者中负载匹配度最高的节点上创

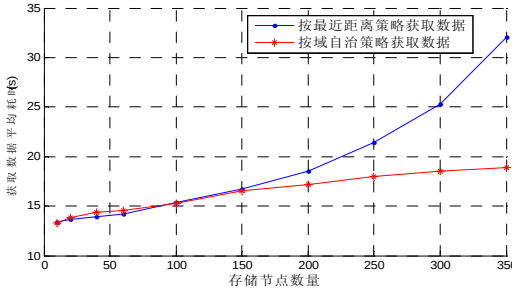
建一个新的副本, 并依照 treap 树的插入规则将新副本插入到二叉树中, 然后重新选取存储节点获取数据.

3 实验仿真与分析

为验证有效性, 我们对上文提出的数据副本管理策略进行了仿真. 仿真硬件系统为 CPU: Pentium®Dual-Core 2.93GHz, 内存: 2GB, Windows7, jdk1.6; 仿真软件系统为 CloudSim; 对比策略包括副本数量固定、位置固定以及按照最近距离获取副本的策略; 仿真指标为服务数据所需的时间. 通过设定 DataCenterBroker 类中获取数据时副本的选择策略, 得到仿真结果如图 3 示:



(a)



(b)

图 3 获取数据策略的对比

在图 3 中, (a)图表示在数据中心节点数量一定的情况下, 两种策略在不同数据请求频率下的效果对比, 由图可知, 随着服务请求并发的增大, 对数据的请求频率不断提高, 与按最近距离策略对比, 本文提出的策略取得的效果越来越明显; (b)图表示对不同的数据获取请求为泊松分布的情况下, 两种策略在数据中心不同节点规模下的效果对比, 由图可知, 在数据中心规模较小的情况下, 使用本文提出的数据副本管理策略, 获取数据速度比按最近距离策略获取数据速度略低, 但随着数据中心规模的增大, 该策略获取数据的速度下降很慢, 可见在数据中心规模很大的情况下, 该策略效果显著.

4 结语

本文针对云计算环境下数据密集型应用中数据存储的副本管理问题, 分析数据获取的相关特性, 采用域自治策略, 对数据副本的选择机制进行改进. 仿真实验结果表明, 采用域自治策略, 并通过负载匹配选择节点, 相比简单的通过距离远近选择节点, 能较快的获取所需数据. 如何针对数据密集型计算的特点, 高效地维护数据副本的一致性, 将是下一步的研究方向.

参考文献

1 Bryant RE. Data intensive supercomputing: The case for DISC. http://www.cs.cmu.edu/~bryant/pubdir/cmu-cs-07-128 pdf(2007.05.10)

2 Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Lee G, Patterson DA, Rabkin A, Stoica I, Zaharia M. Above the Clouds: A Berkeley View of Cloud Computing. [2009-02-10]. http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf

3 http://aws.amazon.com/s3/(2012.05.30).

4 Ghemawat S, Gobioff H, Leung ST. The google file system. SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems principles, ACM New York, NY, USA, 2003:29–43

5 Shvachko K, Hairong K, Radia S, Chansler R. The Hadoop Distributed File System. Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium. 2010:1–10.

6 Fedak G, He H, Cappello F. BitDew: A programmable environment for large-scale data management and distribution. Procc. of the 2008 ACM/IEEE Conference on Supercomputing. Austin, Texas, USA, 2008:1–12.

7 Madathil DK, Thota RB, Paul P, Xie T. A static data placement strategy towards perfect load-balancing for distributed storage clusters. Parallel and Distributed Processing, 2008. IPDPS 2008. 2008:1–8.

8 李俊,吴华鑫,杨坚.分布式服务网络中保证 QoS 的服务路由算法研究.小型微型计算机系统,2010,37(7):1283–1287.

9 章宏灿,薛巍.一种双均衡的集群存储资源映射方法.清华大学学报(自然科学版),2009,49(10):124–127.

10 王雷,董彬如.负载敏感的 P2P 覆盖网.计算机系统应用, 2011,20(12):50–54.