文章编号:1000－2367(2015)04－0138－06 DOI:10.16366/j.cnki.1000－2367.2015.04.026

一种云计算数据副本动态管理策略

李功丽[a,b],赵晓焱[a,b],刘 慧[a,b]

(河南师范大学 a.计算机与信息工程学院;b.智慧商务与物联网技术河南省工程实验室,河南 新乡 453007)

摘 要:为了提高云计算中数据的可用性和缩短响应时间,提出了数据副本的方法.传统的静态副本方法无法适应云计算的动态性特征,针对这个问题提出了动态副本策略.首先,基于用户需求确定副本数目;然后根据局部性原理和资源可用度,确定数据副本的存储位置.最后的实验结果证明了该策略降低了平均响应时间.

关键词:动态副本;云计算;zipf 定律;局部性原理;可用度临界值

中图分类号:TP311 **文献标志码**:A

随着 Internet 技术的飞速发展,出现了越来越多的数据密集型应用,这些应用常常需要对海量的数据进行处理.例如:在生物信息领域的基因序列测序和比对等操作中,动辄上亿个碱基对呈现出 TB(10^{40} B)级的数据量[1];Google 公司每天要处理约 24 PB(10^{50} B)的数据;Facebook 上每天更新的照片量超过 1000 万张[2].传统的存储模式和数据处理方法无法满足这种分布的、大规模的数据处理的需要.

云计算作为一种超大规模的分布式计算系统,使得海量数据处理变得可能.云计算描述了一种新的数据共享与服务共享模式[3],它将 Internet 上存在着的大量分散的、独立的、异构的存储系统组织成一个逻辑意义上的整体,为用户提供高效的、可扩展的、海量的存储资源.所以云计算为海量数据处理提供了有力的支撑.

现在,基于传统 Internet 的云计算技术应用于海量数据处理所面临的主要挑战是如何支持数据快速读取.在云计算中阻碍数据快速读取的主要原因是广域网的高延迟性.副本技术[3]是一种数据管理机制,将数据复制多份分别放在分布式文件系统的多个节点上,这不仅可以提高数据的可靠性和可用性,还可以有效降低数据的网络访问延迟,提高系统的负载均衡性.因为副本技术需要复制多个副本在世界各地的分布式节点上,所以这就需要有效的策略对众多副本进行有效的管理.

副本策略分为静态副本策略和动态副本策略.静态副本策略基于已知的访问方式,在文件创建时确定副本的数目及存储位置.如现在被广泛使用的 Apache 的 HDFS(Hadoop Distributed File System)就采用静态副本管理机制.HDFS 的副本系数默认为 3,通过多个副本来提高数据的可靠性和访问效率[4].然而,HDFS 副本管理策略是在文件分块时自动创建固定数目的副本,该机制难以适应用户访问量的动态变化.如果副本系数过大,则浪费存储空间,且给副本维护带来较大的负担;而如果副本系数设置过小,则不能有效提高系统性能.

动态副本策略根据资源环境的变化而动态调整副本数目和位置,以适应不稳定的资源环境[3].文献[5]对 hadoop 的副本放置策略进行了改进,根据结点网络距离与数据负载计算每个结点的调度评价值,据此选择一个最佳的远程放置结点,但是该策略没有考虑结点性能的差异.文献[6]提出了根据计算结点的处理能力按比例存放数据的策略,该放置策略主要考虑结点异构性.文献[7]提出了一种基于节点效用的副本放置策略,节点效用主要表示节点的可靠性,是节点失效次数和被请求次数的比值.

收稿日期:2014-06-27;**修回日期**:2015-03-17.

基金项目:国家自然科学基金(U1204609);河南省教育厅科学技术研究重点项目(12A520025);河南师范大学青年基金项目(2012QK21).

作者简介(通信作者):李功丽(1981－)女,河南信阳人,河南师范大学讲师,研究方向为分布式系统、云计算,E-mail:4928944@qq.com.

本文针对云计算环境中的副本管理，设计了一种基于用户需求的动态管理策略. 首先根据用户需求，设置合适的副本个数，再利用访问中存在的局部性原理，选择存放副本的目的主机，从而动态调整副本数目并有效提高系统的读写速度和降低响应时间.

2　问题描述及定义

为了建立数据副本动态管理模型，首先使用基于资源位置的资源聚类. 根据资源所在的 IP 网段进行划分，从而将整个云端的资源集合划分为若干个局部域，见图 1. 为了防止单点失效，每个局部域初始时设置两个管理节点，它们负责对局部域中的资源节点进行管理. 而管理域则实现对各个局部域的管理，当系统规模变大，资源节点增多，可以增加管理域中节点的个数，以实现对系统的有效管理，并保证了系统的可扩展性.

定义 1　整个云端的组成用一个元组定义如下：$\text{Cloud} = \{GM, LA_1, LA_2, \cdots, LA_n\}$，这里 GM 指的是全局管理域，LA_i 表示各个局部域.

定义 2　每个局部域的组成定义：$LA_i = \{LM_i, R_{i1}, R_{i2}, \cdots, R_{im}\}$，$LM_i$ 指的是局部域 LA_i 的管理结点，$R_{i1}, \cdots, R_{im}$ 指的是局部域 LA_i 中的各个资源结点.

定义 3　文件目录信息元组：File = {id, size, hkey, dup_num, dup_pointer}，分别表示文件 ID、文件的大小、文件热度值、文件副本数目以及指向保存副本位置的链表的首指针. 文件的热度值采用(1) 式计算.

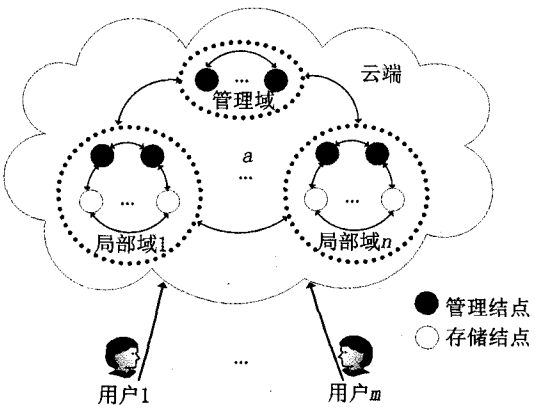

图 1 云端结构图

$$\text{hkey}_{i-k} = \theta * v_{i-k} + (1-\theta)\frac{\sum_{i=1}^{k-1} v_i}{k-1}. \qquad (1)$$

v_k 是指第 k 个时间周期内文件被请求的次数. θ 是一个调节系数，取值范围为[0,1]，如果文件在不同周期访问频率变化较大，可以让 $\theta > 0.5$，而如果文件在各个周期访问次数比较均匀，可以让 $\theta < 0.5$. hkey 表示了该文件的重要程度，用于确定副本的数目和在文件替换时使用.

定义 4　平均响应时间：用户发出请求数据 i 的时间记为 $t_{i,s}$，而最终将数据读取到本地的时间记为 $t_{i,e}$，则该数据的响应时间 $t_{i,r}$ 为：

$$t_{i,r} = t_{i,e} - t_{i,s}. \qquad (2)$$

假设某个周期内系统共有 m 个数据请求，它们的平均响应时间记为 $t_{a,r}$ 为，它的定义如下：

$$t_{a,r} = \frac{1}{m}\sum_{i=1}^{m} t_{i,r}. \qquad (3)$$

3　算法描述

3.1　局部性原理

操作系统中的局部性原理是指：程序在执行时往往具有局部性特征，即在较短时间内，程序执行仅局限于某个部分，相应地它所访问的存储空间也局限于某个区域. 局部性原理同样适用于网络环境下的资源. 对用户来说，下一次要访问的资源访问的与前一次访问的资源在同一个局部域的概率非常大[8]. 文献[9]进一步指出文件访问的局部性特征包括：1) 时间局部性：最近被访问过的文件可能马上被再次访问；2) 地理局部性：也称用户局部性，是指一个用户最近访问的文件可能被与该用户邻近的用户再次访问，即当前访问这个数据文件的用户，在未来一段时间内，它附近的用户仍有可能访问该数据文件. 所以本文首先利用时间局部性原理确定副本数目，再利用空间局部性原理选择副本的放置节点.

3.2　确定副本个数

初始时，数据副本个数均为 2. 在系统运行过程中，根据局部性原理，使用文件的历史访问记录来动态调整副本数目. 文件 j 在第 $k-1$ 周期被请求的次数 $f_{j_{r_{k-1}}}$ 与文件在周期 k 所需要的副本数目 $f_{j_{n_k}}$ 之间的映射

通过公式(4)确定[10]：

$$f_{j_{n_k}} \approx \eta * \sqrt[b]{f^a_{j_{r_{k-1}}}} \tag{4}$$

其中，η 为根据系统性能设置的正常数，a，b 是正整数，且 $b \geq a$.

根据文件历史访问用户的 IP，对每个 IP 执行一次映射，将 IP 映射到网络拓扑中离用户最近的一个局部域，Hash(IP) $= LA_i (1 \leq i \leq n, n$ 为系统中局部域的个数).

假设针对文件 j 在周期 k 共有 m 个请求，则根据上面的法则，会映射产生 m 个结果. 假设这 m 个结果中，有 m_i 个是映射到局部域 LA_i，则 LA_i 中应该存放的副本个数 $f_{j_{n_k}}(LA_i)$ 按照公式(5) 确定：

$$f_{j_{n_k}}(LA_i) \approx \frac{m_i}{m} * f_{j_{n_k}} \tag{5}$$

Adjust ($f_{j_{n_k}}$) − Algorithm// 在周期 k 调整文件 f_j 副本总数的算法

If ($f_{j_{n_k}} - f_{j_{n_{k-1}}} \neq 0$)// 如果 f_j 在周期 k 的副本总数需要调整

For ($i = 1, i <= n, i++$)// 对所有的局部域调用下面的算法

If ($f_{j_{n_k}}(LA_i) >= 1$)// 如果需要在某个局部域中调整副本

Adjust ($f_{j_{n_k}}(LA_i), LA_i$) − Algorithm// 调用局部域中调整副本的算

Endif

Endfor

Endif

而对于在局部域内调整副本的算法 Adjust($f_{j_{n_k}}(LA_i), LA_i$) − Algorithm 如下：

Adjust ($f_{j_{n_k}}(LA_i)$,LA$_i$) − Algorithm// 在周期 k 调整 LA_i 内文件 $f_{j_{n_k}}(LA_i)$ 副本个数算法

$\Omega = f_{j_{n_k}}(LA_i) - f_{j_{n_{k-1}}}(LA_i)$// 将结果与文件已经存储的数据副本进行比较

If ($\Omega < 0$)// 如果需要减少局部域 LA_i 中 f_j 的副本

For($i = -1$, $i >= \Omega$, i——)// 减少的副本个数

Reduce (f_j, LA_i) − Algorithm// 调用把 f_j 的副本从局部域 LA_i 中删除的算法

Endfor

Endif

If ($\Omega > 0$)// 如果需要增加局部域 LA_i 中 f_j 的副本

For($i = 1$, $i <= \Omega$, $i++$)// 增加的副本个数

Add (f_j, LA_i) − Algorithm// 调用在局部域 LA_i 中为 f_j 增加副本的算法

Endfor

Endif

3.3 调整副本

根据公式(4)，可以确定任意局部域中应该保存的某个文件的副本个数，下一步要做的就是在该局部域中选择合适的节点，增加或删除副本. 在确定增加或删除副本的目的主机时，必须要考虑系统的负载平衡问题. 首先定义资源节点可用度，对于局部域 LA_i 中的节点 s_{ix}，它的可用度定义如下：

$$U(s_{ix}) \approx \gamma * s_{ix,b} + (1 - \gamma) * s_{ix,s}. \tag{6}$$

可用度表示了节点当前的性能，其中，$s_{ix,b}$ 表示节点 s_{ix} 的网络带宽，$s_{ix,s}$ 表示节点 s_{ix} 的可用存储空间. 如果需要在局部域中寻找一个节点存放副本，则选择可用度值最大的节点. 在局部域 LA$_i$ 中为数据副本 j 寻找放置节点的算法描述如下：

Add (f_j, LA_i) −Algorithm//在局部域 LA_i 中为数据 f_j 增加副本的算法

If (f_j not in LA_i)//如果局部域中不存在的数据 f_j 的副本

Request (f_j, GMN)//向全局管理节点申请数据块 f_j 的一个副本

Max (U(s_{ix}),x=1,2,…,m)//寻找 LA_i 所有节点中可用度最高的

Copy (f_j, s_{ik})//将数据副本存储在可用度最高的节点 s_{ik} 上

Send (info, LA_i)//向局部管理节点发送信息，以更新文件目录信息

Update U(s_{ik})//更新节点的可用度值

Endif

删除局部域 LA_i 中的数据副本 f_j 的时候，则选择该数据的副本所在的节点中可用度最低的. 算法描述如下：

Reduce (f_j, LA_i)－Algorithm//从局部域 LA_i 中删除 f_j 的数据副本的算法

Search (f_j,LA_i)//寻找 LA_i 中所有存储数据 f_j 的副本的节点

Min (U(s_{iy}), $\exists f_j \in s_{iy}$)//寻找存有 f_j 副本的节点中可用度最低的节点

Delete (f_j, s_{ik})//删除节点 s_{ik} 上 f_j 的副本

Send (info, LA_i)//向局部管理节点发送信息，以更新文件目录信息

Update U(s_{ik})//更新节点的可用度值

因为每次增副本时，都是放置在局部域中可用度值最高的节点上，从而可以在增加数据副本之后，节点仍然正常运行. 而删除副本的时候，选择所有存储该副本的节点中可用度最低的进行删除，这样可以改善可能存在的过载节点的性能.

3.4 可用度临界值

如果在放置副本时，搜索到的可用度最高的节点当前是过载的，则意味着该局部域当前整体是过载的，此时需要增加该局部域中的资源节点数目，以避免系统因为处于超负荷运行状态所导致的系统性能下降. 而如果在删除副本 F 之后，局部域中存有该副本的节点中可用度最小的节点却大于某个临界值，这说明系统整体处于轻载状态，此时可以适当关闭一些资源主机，从而节省系统的能源消耗. 所以根据节点的性能设置可用度的最小值 $u(s_{ik})_{\min}$ 和最大值 $u(s_{ik})_{\max}$ 对局部域中节点状态进行判断.

Threshold (s_{ik})－Algorithm//判断节点 s_{ik} 的可度用是否超过临界值

If ($\max(u(s_{ik})) < u(s_{ik})_{\min}$, $1 \le k \le m$, m 是 LA_i 中的节点个数) //如果节点的最大可用度小于最小临界值

Add(s_{ix}, LA_i)//在该局部域中增加资源节点 s_{ix}

Endif

If ($\min(u(s_{ik})) > u(s_{ik})_{\max}$, $1 \le k \le m$, m 是 LA_i 中的节点个数)//如果节点中可用度最小的大于最大临界值

For($\forall f_j \in s_{ik}$)//把节点 s_{ik} 上的所有副本迁移到局部域 LA_i 的其它节点上

transfer(f_j, LA_i)

turnoff(s_{ik}, LA_i)//关闭该资源节点

Endfor

Endif

所以每次在节点上放置副本之前，可以先调用 *Threshold()_Algorithm* 判断节点的状态，这样即可以在节点过载时，主动增加资源节点，保证系统的负载平衡，又可以在系统整体空闲时，适当关闭节点，从而减少能源消耗，提高系统资源的利用率.

4 仿真实验与结果分析

本文使用云仿真平台 CloudSim[11] 对本文的动态副本策略法进行仿真实验. 在 CloudSim 基本类的基础上编写相应的继承类，并对 CloudSim 进行扩展.

设置系统共有 5 个局部域，局部域之间连接带宽为 2 MB/s，局部域内的各资源节点之间的连接带宽设置为 10 MB/s. 磁盘的传输速度为 20 MB/s，文件大小为 200 M. 初始时，每个节点的可用存储空间为 10 G 到 100 G 之间的随机数，系统共有 100 个数据文件，按照默认的副本数为 2 分布在不同的局部域中.

在实验中系统一共生成 1000 个用户请求，请求在第 k 个周期到达的概率 p_k 服从. $\lambda=3$ 的泊松分布

$$p_k = \frac{\lambda^k}{k!} * e^{-\lambda}. \tag{7}$$

在一个连续的时间段内,对某站点的 100 个文件进行了监测,文件的访问概率如图 2 所示.

图 2 说明系统对于文件的访问并不是均匀的,而是集中在排名前 20%的文件,这也符合前面描述的文件访问的空间局部性原理. 所以在实验中,动态副本调整算法只针对热度值在前 20%的文件进行,而对于其余的文件,使用默认的副本数目 2. 且这 2 个副本分别存放在不同的局部域中,以降低文件失效的概率.

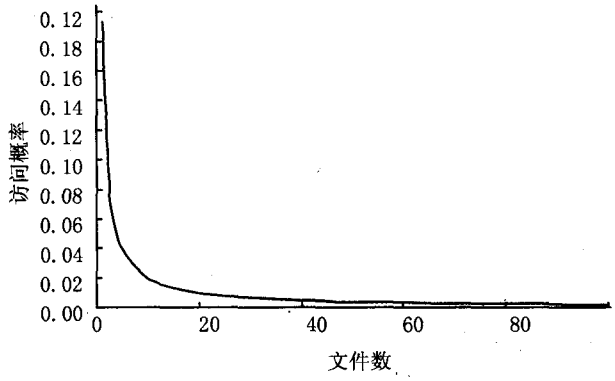

图 2 文件的访问概率曲线图

实验中为了量化计算,针对每个文件的请求概率使用 like zipf 分布[12]:

$$z_i = \frac{C}{i^a}, \tag{8}$$

其中,i 是针对文件 i 的请求个数在所有文件中的排名,这里用热度值排名替代. a 取值为 1,常数 C 满足

$$C * \sum_{i=1}^{m} \frac{1}{i} = 1. \tag{9}$$

所以在文件个数为 100 时,$C = 0.1928$,根据公式(8)和公式(1)所计算出来的文件热度值,可以计算出针对文件 i 在周期 k 内的请求个数 $f_{i_{r_k}}$:

$$f_{i_{r_k}} \approx \text{sum}_{req} * p_k * z_i, \tag{10}$$

这里 sum_{req} 是实验过程中请求的总数,把请求个数 $f_{i_{r_k}}$ 使用公式(4)进行映射,即可得到该文件在周期 k 的副本个数 $f_{i_{n_k}}$,再使用公式(5),即可计算得到该文件在某个局部域中的副本个数 $f_{i_{n_k}}(LA_i)$.

实验中,取 $a = 1, b = 2, \eta = 10$. 在 10 个周期中(每个周期为 300 s),根据本文的动态副本策略,100 个文件的副本的总数如图 3 所示.

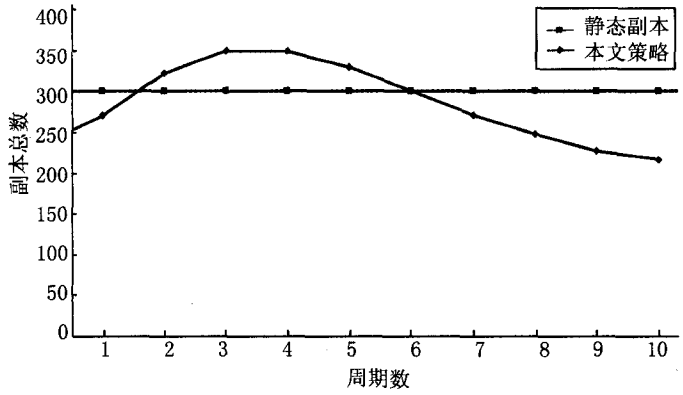

图 3 不同策略副本总数对比图

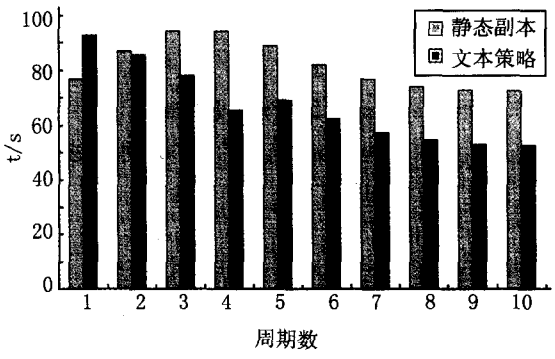

图 4 不同策略的响应时间对比图

根据用户在不同周期到达的请求,计算得到的在前 10 个周期,本文动态算法的响应时间与默认的副本数 3 的响应时间对比如图 4 所示.

图 4 的结果显示,在第一个周期,因为没有历史用户请求数据,所以本文算法就等价于副本数为 2 的静态副本策略,所以它的平均响应时间明显大于副本数为 3 的静态副本策略. 而在后面的周期中,本文的动态副本算法因为考虑到了(1)用户访问的局部性特征;(2)存储副本的节点与用户之间的网络距离,所以整体的平均响应时间均短于静态副本策略,从而获得了较好的服务性能,改善了云计算环境中数据传输延迟大、响应时间长的问题.

5 结 语

传统的静态副本策略是在初始时创建固定数目的副本，且在选择存储副本的主机时是随机的，没有考虑用户与副本位置的关系，这导致数据传输延迟大、平均影响时间较长. 本文基于云计算环境的特点，设计了基于资源位置聚类的资源管理模型. 然后利用用户访问的局部性特征，设计了动态副本管理策略. 该策略考虑数据存储主机节点和用户之间的位置关系，使用可用度衡量节点的性能，选择性能最优的节点放置副本，从而在保证系统负载平衡的同时，降低了平均响应时间. 但是本文的动态副本策略是基于历史数据的，所以存在一定的滞后，如果请求数目随时间变化较大，本文的算法性能可能会降低，所以未来进一步的工作重点是研究文件访问的特点和规律，使用有效的预测算法，解决滞后的问题.

参 考 文 献

[1] 罗浩宇，陈旺虎. 基于社会网络特征的云服务副本放置策略[J]. 计算机应用，2013，33(8)：2143-2146.

[2] Viker M S, Kenneth C. Big Data: A Revolution That Will Transform How We Live, Work, and Think[M]. Houghton: Mifflin Harcourt, 2012: 28.

[3] XU Jing, YANG Shoubao, WANG Shuling, et al. CDRS: An adaptive cost-driven replication strategy in cloud storage[J]. Journal of the Graduate School of the Chinese Academy of Sciences, 2011, 28(6): 759-767

[4] 陈　康，郑纬民. 云计算：系统实例与研究现状[J]. 软件学报，2009，20(5)：1337-1348.

[5] 林伟伟. 一种改进的 Hadoop 数据放置策略[J]. 华南理工大学学报：自然科学版，2012，40(1)：152-158.

[6] XIE Jiong, YIN Shu, RUAN Xiaojun, et al. Improving map-reduce performance through data placement in heterogeneous hadoop clusters[C]. IPDPS Workshops, Atlanta, 2010.

[7] Ben Charrada, Ounelli F, Chettaoui. An efficient replication strategy for dynamic data grids[C]. Proceedings of International Conference on P2P Parallel Grid Cloud and Internet Computing, Washington D C, 2010.

[8] 瞿　中，邱玉辉. Hash 函数实现数据包分流算法研究[J]. 计算机科学，2006，33(2)：67-70.

[9] 田　田，罗军舟，宋爱波，等. 网格虚拟组织副本协作预取机制[J]. 软件学报，2011，22(10)：2372-2384.

[10] 夏前军，李庆虎，叶晓俊. 基于 GFS-Net 的动态复制[J]. 计算机科学，2005，32(8)：67-69.

[11] Calheiros R N, Ranjan R, de Rose C A F, et al. Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services[J]. arXiv preprint arXiv: 0903. 2525, 2009.

[12] Leung Y W, Hou R Y T. Assignment of movies to heterogeneous video servers[J]. IEEE Transactions on Systems, Man, and Cybernetics Part A: System and Human, 2005, 35(5): 665-681.

Dynamic Management Strategy for Data Replications Based on Cloud Computing

LI Gongli[a,b], ZHAO Xiaoyan[a,b], LIU Hui[a,b]

(a. College of Computer & Information Engineering, b. Engineering Laboratory of Intellectual Business and Internet of Things Technologies, Henan Normal University, Xinxiang 453007, China)

Abstract: In order to enhance the availability and reduce the response time of data in the cloud computing, we present the data replication scheme. The traditional static replication methods can't adapt to the dynamic characteristics of cloud computing. For this issue, we propose the dynamic replications strategy. Firstly, we compute the number of replication based of the user's demand, and then choose the storage location for data replication according to the resource availability function and locality principle. The simulation experiment proves that the dynamic data replication strategy reduces the average response time.

Keywords: dynamic replication; cloud computing; zipfs law; locality principle; availability threshold