一种基于动态修正值的分布数据流 Top-K查询处理算法

刘维弋 金远平
(东南大学计算机科学与工程学院 江苏 南京 210096)

摘 要 对分布式数据流进行查询,得到数值最大的 K个对象(Top-K观测查询),最直接的解决方法是由中心结点处理分布式数据流,但这种方法导致中心结点和网络负载较大。提出一种基于动态修正值的查询算法,通过对观测数据进行计算得到修正值,并利用该修正值对不同结点处的对象数据进行操作,从而无需将结点数据流全部发送到中心结点就能完成 Top-K观测查询。因而可以减少对网络带宽的要求和降低中心结点的负载,同时还能保持查询结果的完全准确。

关键词 分布式数据流 Top-K观测查询 网络负载 动态修正值

A DYNAMIC ADJUSTMENT FACTOR BASED ALGORITHM FOR TOP-K QUERIES

Liu Weiyi Jin Yuanping
(School of Computer Science and Engineering, Southeast University, Nanjing 210096, Jiangsu, China)

Abstract The K largest values can be obtained by querying the distributed data streams (Top-K monitoring queries). The straightforward method is to process the data streams on a coordinator. However, this leads to heavy loads on the coordinator and the network. A dynamic adjustment factor based algorithm for Top-K monitoring queries is proposed. The overall communication cost and the load of the coordinator can be reduced significantly, and at the same time the preciseness of the query results is fully guaranteed.

Keywords Distributed data streams Top-K monitoring queries Load of network Dynamic adjustment factor

1 Top-K查询

对分布式数据流进行实时统计分析具有广泛的应用,如网络检测、感应器观测等。在这些应用中,人们对大量对象进行观测,并利用观测数据进行分析和控制[1]。例如,一家水果销售总公司在城市的不同地点设有 m个水果销售点,每个水果销售点都同时销售同样的 f种水果。为了对水果的采购和分配进行统一调度和控制,总公司需要对这 f种水果的销售数据进行实时监控。在对象众多的实际应用中,人们不可能也不需要对全体对象都进行实时控制,所关心的对象只是它的一个子集。因此我们只需以某种排序标准为依据对分布式数据流进行分析,从而查询出在排序结果中位于最前列的 K个对象(分布数据流 Top-K查询),并进行控制。分布数据流 Top-K查询,即通过对分布式数据流的查询,返回具有最大数值的 k个对象。在上例中,水果销售总公司只需实时查询全市范围内销售量最高的 k种水果,以优化资源配置,保证这 k种水果的供应。

这里需要解决分布式数据流的 Top-K观测查询问题。要得到全体对象的 Top-K子集,最直接的方法就是建立一个中心结点,先将各分布结点的对象观测数据统一发送到中心结点,然后由中心结点求解出 Top-K查询结果。但这种方法面临着资源过度消耗问题。第一,随着观测结点的增多,中心结点由于受到网络通信能力的限制(如网络带宽),无法实时取得各观测结点的数据,因而也就无法进行相应的实时数据统计。第二,这种方法需要所有分布结点同时向中心结点发送数据,导致中心结点周围的网络产生大量数据流,从而占用较多的网络资源,并有可能造成网络拥塞。第三,如果连续多次查询的 Top-K结果相同,则以上资源消耗是可以通过计算来避免的。

2 Top-K查询相关的研究

目前对于分布式 Top-K查询的研究有很多,其中较为典型的有文献[2-6]。文献[5]虽然也讨论 Top-K问题,但重点研究 P2P网络中结点数据的分布式查询。文献[6]研究的是关系数据库中的 Top-K问题。文献[2,3]都是对远端分布结点的数据源进行一次性 Top-K查询。文献[2]的研究重点是如何提供精确的 Top-K查询结果,而文献[3]同时研究了提供精确和粗略的两种 Top-K查询结果的方法。由于没有提供有效的机制来跟踪 Top-K的变化,文献[2,3]的方法都不适合实时观测 Top-K查询的应用领域。文献[4]提出的分布式数据流 Top-K观测查询处理方法利用修正值使同一个对象在不同结点处观测数据的分布趋于平衡。

在文献[4]利用修正值 δ对数据进行约束的过程中,我们注意到 δ值在同一次约束规则中保持不变。因此利用这种算法得到的修正值不能动态地适应数据变化趋势,也就无法充分发挥平衡数据分布的作用。例如:有两个分布结点 N_1、N_2 和一个中心结点 N_0。N_1 的观测数据为:对象 1: $V_{1,1}=8$,对象 2: $V_{2,1}=1$; N_2 的观测数据为:对象 1: $V_{1,2}=2$,对象 2: $V_{2,2}=3$。在全局范

收稿日期:2007-03-14。国家自然科学基金重大研究资助项目(90412014)和国家自然科学基金项目(60603040)。刘维弋,硕士,主研领域:数据库。

围内进行统计计算,对象 1的值为 $V_{1,1}+V_{1,2}=10$,对象 2的值为 $V_{2,1}+V_{2,2}=4$。因为 $10>4$,所以对象 1的值最大,得到全局结果 Top-K = {对象 1} (K = 1)。在文献[4]中,修正值可以使观测对象的数据在不同结点处重新分配,从而保证每个分布结点处的 Top-K保持一致。同一个对象在不同结点处的修正值的和应为 0,以保证数据的正确性。如上例,在结点 2处,对象 1值小于对象 2值,所以结点 2局部处的 Top-K = {对象 2},这个局部结果与全局结果 Top-K = {对象 1}不一致。将对象 1在结点 1和结点 2处的值分别与其对应的修正值相加 $V_{1,1}+\delta_{1,1}=5$ $V_{1,2}+\delta_{1,2}=5$ (假设结点 1上对象 1的修正值 $\delta_{1,1}=-3$,结点 2上对象 1的修正值 $\delta_{1,2}=3$,结点 0上对象 1的修正值 $\delta_{1,0}=0$,且满足 $\delta_{1,1}+\delta_{1,2}+\delta_{1,0}=0$),即对象 1在结点 1处的数据减小 3变为 5,同时在结点 2处的数据增大 3变为 5,相当于对象 1的数据在全局范围内没发生变化,只是对象 1在结点 1处的一部分数据通过修正值被分配到结点 2(修正值为正表示向该对象值添加数据,修正值为负表示从该对象值中取出数据)。通过引入修正值计算,结点 1处对象 1大于对象 2,结点 2处对象 1大于对象 2,所以无论在结点局部范围还是全局范围, Top-K结果均为{对象 1}。但此方法计算出的修正值仍有不足的地方,准确利用文献[4]中提供的算法计算出结点 1上对象 1的修正值为 $\delta_{1,1}=-5$,对象 2的修正值 $\delta_{2,1}=0$。同理,其他结点的修正值为: $\delta_{1,2}=3, \delta_{2,2}=0$; $\delta_{1,0}=2, \delta_{2,0}=0$。下次处理 Top-K查询时,假设新观测数据为 $V_{1,1}=9, V_{2,1}=9$; $V_{1,2}=10, V_{2,2}=4$,将观测值与修正值相加则得到修正后的数据,如表 1所示。

表 1 利用算法[4]进行 Top-K计算的所有相关数据

物理位置	结点 1		结点 2	
观测对象	对象 1 $V_{1,1}$	对象 2 $V_{2,1}$	对象 1 $V_{1,2}$	对象 2 $V_{2,2}$
第 1次观测值	8	1	2	3
第 2次观测值	9	9	10	4
修正值[4]	$\delta_{1,1}=-5$	$\delta_{2,1}=0$	$\delta_{1,2}=3$	$\delta_{2,2}=0$
修正后的数据 =第 2次观测值 +修正值	4	9	13	4
局部 Top-K	结点 1局部的 $Top\text{-}K^1$ = {对象 2}		结点 2局部的 $Top\text{-}K^2$ = {对象 1}	
全局 Top-K	上次观测的全局 Top-K =本次观测的全局 Top-K = {对象 1}			

从表 1的计算结果中可以看出,结点 1局部的 $Top\text{-}K^1$ = {对象 2} (上标“1”表示在结点 1处进行 Top-K查询的局部结果)。由于 $Top\text{-}K^1 \neq$上次观测的全局 Top-K,从而引起中心结点同时向各分布结点通信以取得各对象在各结点处的新观测值,即全网通信,并进行新一轮的中心结点 Top-K计算。但事实上计算结果仍为 Top-K = {对象 1}。这提示我们,本次全网通信是有可能避免的。

3 本文的工作

通过对上述问题的深入分析和研究,本文提出一种基于动态修正值的 Top-K查询算法。本算法依然采用变量 δ这个名称,但 δ值的计算方法与文献[4]不同,它的产生和更新完全以观测数据为基础,因而本算法能够动态准确地描述数据变化趋势。修正值 δ中存放对象数据与 Top-K边界值的差值被重新平衡分配后的结果。Top-K边界值,指对象在 Top-K结果中所要求的最小的数据值。如上面的例子,观测数据为: $V_{1,1}=8, V_{2,1}=1$; $V_{1,2}=2, V_{2,2}=3$。对象 1在全局范围内的观测数据之和为 $V_{1,1}+V_{1,2}=10$,对象 2的观测数据之和为 $V_{2,1}+V_{2,2}=4$。全局结果 Top-K = {对象 1},对象 2是不在 Top-K中的值最大的对象,所以取 Top-K边界值为 4 + 1 = 5 (计算公式详见第 2节)。因此对象 1的修正值之和应为 10 - 5 = 5,表示对象 1的数据在减少 5后,依然能保证对象 1在 Top-K中。当新的观测数据到来后,将对象 1的修正值之和 5按照规定的策略分配到各分布结点处,通过计算,对象 1在结点 1处的修正值为 8,在结点 2处的修正值为 - 6,在结点 0处修正值为 3 (保证对象 1的修正值之和仍为 5,算法详见第 4节)。这样可以将多余的数据 5应用到下一次观测数据的 Top-K计算中,若下次对象 1的观测数据过小或过大,则用修正值与其相加,以尽量保证对象 1仍然在新的 Top-K查询结果中。需要说明的是,非 Top-K结果中的一个对象用此方法计算出的修正值之和为负值,这表示暂借数据。当下次观测数据过大时,可以通过修正值减小对象数据,以尽量保证非 Top-K结果中的对象在下次观测计算中仍然在 Top-K之外。下面还用上述例子来说明本算法相对于文献[4]的算法的优越性,如表 2所示。

表 2 利用本文算法进行 Top-K计算的所有相关数据

物理位置	结点 1		结点 2	
观测对象	对象 1 $V_{1,1}$	对象 2 $V_{2,1}$	对象 1 $V_{1,2}$	对象 2 $V_{2,2}$
第 1次观测值	8	1	2	3
第 2次观测值	9	9	10	4
变化值 =第 2次观测值 - 第 1次观测值	$\triangle_{1,1}=1$	$\triangle_{2,1}=8$	$\triangle_{1,2}=8$	$\triangle_{2,2}=1$
修正值	$\delta_{1,1}=8$	$\delta_{2,1}=0$	$\delta_{1,2}=-6$	$\delta_{2,2}=0$
修正后的变化值 =变化值 +修正值	9	8	2	1
局部 Top-K	结点 1局部的 $Top\text{-}K^1$ = {对象 1}		结点 2局部的 $Top\text{-}K^2$ = {对象 1}	
全局 Top-K	上次观测的全局 Top-K = 本次观测的全局 Top-K = {对象 1}			

本算法可以有效减少资源消耗:

(1) 在单个分布结点的局部范围内对数据进行处理以避免没有必要的网络通信,从而减轻网络负载。

(2) 单个分布结点确实无法处理本地 Top-K数据时,它才会要求中心结点参与处理。此过程只涉及到一个分布结点和中心结点,因而降低了中心结点的负载。

4 算法详解

4.1 定义及说明

局部 Top-K:在分布结点范围内进行计算,从而得到的值最大的 K个对象。全局 Top-K:计算对象在各分布结点处的值的和,并求出值最大的 K个对象。旧 Top-K:本次查询发生前,已知的 Top-K结果。新 Top-K:本次查询发生后,新计算出的 Top-K结果。失效:即 Top-K失效,指在新一轮计算后,得到的新 Top-K与旧 Top-K不一致。有效:定义与失效相反。失效集合

P_j:指在编号为 j的分布结点上产生失效后,引起失效的对象集合,即 Pj = [{新局部 Top-K} - {旧局部 Top-K}] ∪ [{旧局部 Top-K} - {新局部 Top-K}] = [{新局部 Top-K} ∪ {旧局部 Top-K}] - [{新局部 Top-K} ∩ {旧局部 Top-K}]。O_i:表示要观测的对象,共有 $O_1, O_2, O_3, \cdots, O_f$, f个被观测对象。$n_j$:代表观测对象的结点,共有 $n_1, n_2, n_3, \cdots, n_m$, m个观测结点; n_0 为中心结点。三元组 $<O_i, n_j, V>$:表示在新一轮观测后, n_j 结点上对象 O_i 的所有增加量的累加值 V,简记为 $V_{i,j}$,其中 i表示被观测对象的编号, j表示观测结点的编号。三元组 $<O_i, n_j, \triangle>$:表示在新一轮观测后, n_j 结点上对象 O_i 的观测值比上一次观测值变化了 △,简记为 $\triangle_{i,j}$,其中 i表示被观测对象的编号, j表示观测结点的编号。三元组 $<O_i, n_j, \delta>$:表示在新一轮观测后, n_j 结点上对象 O_i 对应的修正值 δ,简记为 $\delta_{i,j}$,其中 i表示被观测对象的编号, j表示观测结点的编号。二元组 $<n_j, B>$:表示在某次观测后, n_j 结点上的所有对象数据集的 Top-K下界,简记为 B_j,其中 j表示观测结点的编号。B_j 的计算方法, B_j 是编号为 j的结点上的所有经过修正后的对象观测值 $\triangle_{i,j} + \delta_{i,j}$ 中的第 k + 1个大的值。

4.2 算法流程

系统初始化 (图 1给出了算法流程图):

(1)系统开启 ($V_{i,j} = 0$),进入状态 Ⅰ,各分布结点进行第一次观测并得到初始 $V_{i,j}$。

(2)系统进入状态 Ⅳ,中心结点首先通知各分布结点向它发送 $V_{i,j}$ 值,然后计算出全局 Top-K和修正值 $\delta_{i,j}$,最后将结果全局 Top-K和 $\delta_{i,j}$ 发送到各对应的结点 j。

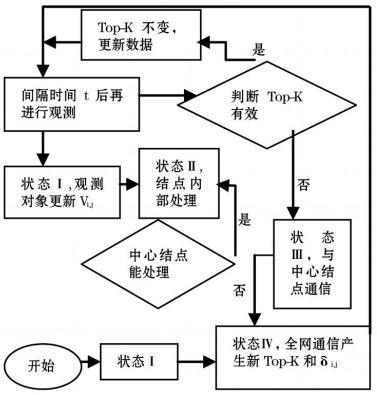

图 1　算法流程图

系统正常运行:

(1)观测时间间隔,各分布结点进入状态 Ⅰ,观测新变化值 $\triangle_{i,j}$,并更新 $V_{i,j}$。

(2)各分布结点进入状态 Ⅱ,进行结点内部数据处理,判断 Top-K是否有效。

若有效:则更新 $\delta_{i,j}$,查询结束, Top-K不变。

若无效:则进入状态 Ⅲ,中心结点与失效结点单独进行通信。如果中心结点能够处理,则将 $\delta_{i,0}$ 值补充分配到失效结点的失效对象上,算法再转向 (2)运行。否则系统进入状态 Ⅳ。

(3)系统进入状态 Ⅳ,先进行全网通信,取得所有数据后中心结点重新计算全局 Top-K和 $\delta_{i,j}$,并将其发送到对应的结点 j。这种情况并不是每次运行到状态 Ⅲ都会发生。只有在状态 Ⅲ中,中心结点也无法处理分布结点的失效时才进入状态 Ⅳ。

4.3 算法详解 (修正值 δ的计算方法与使用)

假设各分布结点每隔时间 t统一进行一次 Top-K查询,各分布结点进入状态 Ⅰ。

状态 Ⅰ:分布结点 j($j = 1, 2, \cdots, m$)对对象 i($i = 1, 2, \cdots, f$)。进行一次观测,从而获得各对象在各分布结点的变化值 $\triangle_{i,j}$,在结点 j上更新 $V_{i,j}$ ($V_{i,j} = V_{i,j} + \triangle_{i,j}$)。之后,各结点进入状态 Ⅱ。

状态 Ⅱ:修正 $\triangle_{i,j}$ ($\triangle_{i,j} = \triangle_{i,j} + \delta_{i,j}$),然后按降序排列 $\triangle_{i,j}$,判断局部 Top-K的有效性。

有效,则计算 B_j (见 4.1),对 $\delta_{i,j}$ 进行操作,若 $i \in$ Top-K, $\delta_{i,j} = \triangle_{i,j} - B_j - 1$;若 $i \notin$ Top-K, $\delta_{i,j} = \triangle_{i,j} - B_j$。当结果 $|\delta_{i,j}| \geq \beta$ 时,则将 $\delta_{i,j}/2$ 发送到中心结点 (β为设定的阀值)。中心结点更新 $\delta_{i,0} = \delta_{i,0} + \delta_{i,j}/2$ 本地结点更新 $\delta_{i,j} = \delta_{i,j}/2$。查询结束, Top-K不变。

失效,则计算出失效集合 P_j (见 4.1),并将其发送到中心结点,转向状态 Ⅲ。

状态 Ⅲ:中心结点与失效结点单独通信。

假设:由于同一次统计中失效的结点可能有多个,则定义失效结点集合为 N,大小为 h。

中心结点先收到各分布结点的失效集合 P_j, $j \in N$。

当 $|\delta_{i,0}| \leq \alpha$ ($j \in N$, $i \in P_j$)时,则转向状态 Ⅳ进行全网通信 (α为设定的阈值)。

逐个分配 $\delta_{i,j} = \delta_{i,0} * \theta_j$, $\delta_{i,0} = \delta_{i,0} * (1 - \theta_j)$, ($j \in N$, $i \in P_j$)。

将新得到的 $\delta_{i,j}$ 发送到各失效结点 j,各分布结点进入状态 Ⅱ。

状态 Ⅳ:全网通信状态。

进行全网通信,中心结点通知分布结点将 $V_{i,j}$ 发送到中心结点。

中心结点计算 $V_{i,*} = V_{i,1} + V_{i,2} + V_{i,3} + \cdots + V_{i,m}$ ($i = 1, 2, \cdots, f$),然后按降序排列 $V_{i,*}$,得到全局 Top-K集合,全局由 Top-K得到每个结点 j的下界对象 O_{Bj}。

计算 $V_{OB*} = V_{OB1} + V_{OB2} + V_{OB3} + \cdots + V_{OBm}$;

计算 $\delta_{i,*} = V_{i,*} - V_{OB*} - m$, $i \in$ Top-K; $\delta_{i,*} = V_{i,*} - V_{OB*}$, $i \notin$ Top-K。

设 F_j ($j = 0, 1, 2, \cdots, m$), $F_0 = 0.5$, $F_j = 0.5/m$ ($j = 1, 2, \cdots, m$)。

分配 $\delta_{i,*}$ 值, $\delta_{i,j} = \delta_{i,*} * F_j$,将计算得到的 $\delta_{i,j}$ 和 Top-K分别发送到分布结点 j。

5 实验与分析

本实验根据水果销售的情况产生模拟数据。假定存在 10台分布于不同地理位置的水果销售终端,他们每分钟各自记录本地 60种水果的销售量,持续时间为 1小时,即每个终端各产生 60条销售记录。所以本实验共生成 600条记录,共 36000个数据。图 2为某个终端的某一条销售记录。水果编号的规则为:将销售商认为销售量可能最大的水果编号在 0到 60的中间部位,并按估计的销售量大小编号。

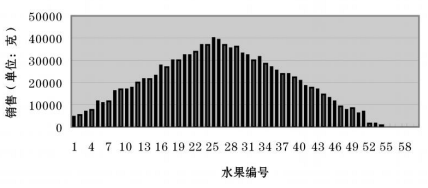

图 2　某个终端的某一条销售记录数据图

5.1 比较方案与评价指标

说明:将文献[4]所提供的算法命名为 SAF_Top-K(Static Adjustment Factor),本文算法命名为 DAF_Top-K(Dynamic Adjustment Factor)。

评价指标:全网通信次数,即中心结点要各分布结点同时向其传送对象观测数据的次数。

影响评价指标的因素有参数变量 α、β、F_j 和 θ_j,为体现实验的公平一致性,本文将 SAF_Top-K与 DAF_Top-K所共有的参数设定为相同固定值,即 $\alpha = 6000$, $F_0 = 0.5$, $F_1 + F_2 + F_3 + \cdots + F_m = 0.5$[4], $\theta_j = 1/5$。所以只剩下 β为评价指标的影响因素。

5.2 数据分析

实验结果如图 3所示。由于 SAF_Top-K中没有 β参数,所以对于同样数据,其参数固定的情况下,SAF_Top-K全网通信次数曲线应为直线,恒为 25。

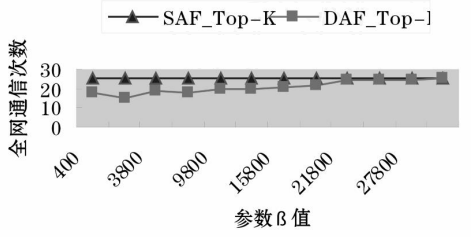

图 3 参数 β值对全局通信次数的影响

图 3说明 DAF_Top-K比 SAF_Top-K进一步降低了全网通信次数,当 β值取 800时全网通信次数取得最小值 15,比 SAF_Top-K节约了 40%的全网通信;当 β值继续变大时(步长为 3000),全网通信次数增大,并逐渐与 SAF_Top-K的全网通信次数曲线接近。

通过进一步实验,DAF_Top-K的单独点对点通信次数比 SAF_Top-K多,大致为 SAF_Top-K的 2倍。但当 β值继续增大时,DAF_Top-K的单独点对点通信次数曲线逐渐与 SAF_Top-K的曲线重合,例如 β = 600000时,SAF_Top-K与 DAF_Top-K的单独点对点通信次数均为 270次。

综合分析,运行 DAF_Top-K算法比 SAF_Top-K算法最多降低大约 40%的全网通信次数,同时 DAF_Top-K的点对点通信次数变大。这说明 DAF_Top-K是将某一些全网通信量分散为单独点对点通信量,虽然单独点对点通信次数增多了,但多个结点同时向中心结点发送数据的情况(全网通信)减少,即 DAF_Top-K算法将集中在同一个时刻的网络通信量均匀地分布在时间轴上,从而避免在某一时刻整个网络因通信量过大而拥塞。DAF_Top-K不仅降低了中心处理机的负载,而且还减少了全网通信次数,从而避免了发生网络拥塞的可能性(全网通信可能会引起中心结点周边网络的拥塞)。

6 总 结

本文提出了一种分布数据流 Top-K观测查询处理的新算法,利用修正值 δ对各分布结点的观测数据进行修正,目标是平滑观测数据的变化以避免不必要的全网通信,因而能优化修正值 δ的修正效果,以避免不必要的资源浪费。

参 考 文 献

[1] Motwani R, Widom J, Arasu A, et al. Query Processing, Resource Management, and Approximation in a Data Stream Management System[C]. In Proc. CIDR, 2003.

[2] Marian A E, Bruno N and Gravano L. Evaluating Top-k Queries Over Web-Accessible Databases[C]. ACM, 2004.

[3] Fagin R, Lotem A, Naor M. Optimal Aggregation Algorithms for Middleware[C]. In Proc. PODS, 2001.

[4] Babcock B, Olston C. Distributed Top-K Monitoring[C]. SIGMOD, 2003.

[5] Balkel W, Nejdl W, Siberski W, et al. Progressive Distributed Top-k Retrieval in Peer-to-Peer Networks[J]. IEEE, 2005.

[6] Bruno N, Chaudhuri S, Gravano L. Top-k Selection Queries over Relational Databases: Mapping Strategies and Performance Evaluation[C]. ACM, 2002.

(上接第 8页)

5 总 结

网络环境下的分布式存储系统中,数据能否在结点间进行有效的放置是保证系统可用性和可靠性所需考虑的最重要问题。数据放置策略的设计需要根据实际应用的环境和网络规模进行综合的考虑。本文总结了分布式存储系统中数据放置所面临的几个主要问题,在评价策略好坏的两个因素(可靠性的恢复开销和系统容错能力)的基础上,讨论了目前系统中主要使用的三种数据放置策略,对其优缺点进行了分析,并基于 Gossip算法的主要思想提出一种基于存储转发的随机放置策略,有效地解决了在动态网络环境下数据副本的放置问题,大大减少了了副本冗余度对系统性能的影响。另外,本文通过采用存储转发随机放置策略的 Turtle系统测试实际数据,验证了该策略的有效性,为其在系统中实际应用提供了可靠的依据。

参 考 文 献

[1] Yin Fu Huang, Jyh Her Chen. Fragment Allocation in Distributed Database Design. Journal of Information Science and Engineering, 2001, 17(5): 491-506.

[2] Qiao Lian, Wei Chen, Zheng Zhang. On the Impact of Replica Placement to the Reliability of Distributed Brick Storage Systems, in Proc. 25th IEEE International Conference on Distributed Computing Systems, June 2005: 187-196.

[3] Lee E, Thekkath C. Petal: Distributed Virtual Disks. ACM SIGOPS Operating Systems Review, 1996, 30(5): 84-92.

[4] John Kubiatowicz, David Bindel, et al. OceanStore: An Architecture for Global-Scale Persistent Storage, in Proceedings of the Ninth international Conference on Architectural Support for Programming Languages and Operating Systems, November 2000: 190-201.

[5] Ghemawat S, Gobioff H, Leung S T. The Google File System, in Proc. of the 19th ACM Symposium on Operating System Principles, Oct 2003.

[6] Lin M J, Marzullo K. Directional gossip: Gossip in a wide area network. Technical Report CS1999 - 0622, University of California, San Diego, Computer Science and Engineering, June 1999.