一种支持负载均衡的存储调度算法

颜秉珩[1]，钱德沛[1,2]
（1. 西安交通大学电子与信息工程学院，710049，西安；2. 北京航空航天大学计算机学院，100083，北京）

摘要：针对应用层存储聚合中的调度问题，提出了一种支持负载均衡的存储调度(LBS)算法. LBS是一种基于策略的调度算法，它将应用对存储资源的需求转换为一系列约束，再通过分析约束之间的关系选择合适的存储节点或者已有的调度方案，从而提高了调度方案的复用率，维护了策略复用与节点负载之间的平衡关系，寻找到最佳的负载均衡策略. 模拟测试表明，LBS算法在负载均衡方面和策略耦合方面明显优于Least和Random算法，负载均衡指标最高可提升10倍左右.

关键词：应用层存储聚合；调度算法；负载均衡
中图分类号：TP393 **文献标志码**：A **文章编号**：0253-987X(2009)10-0061-05

A Scheduling Algorithm for Load Balance Sensitive Storage

YAN Bingheng[1], QIAN Depei [1,2]
(1. School of Electronics and Information Engineering, Xi′an Jiaotong University, Xi′an 710049, China;
2. School of Computer Science, Beihang University, Beijing 100083, China)

Abstract: According to the scheduling problem in storage aggregating of application-level, a scheduling algorithm for load balance sensitive storage (LBS) is proposed based on policy. In the LBS algorithm, requirements from applications for storage are represented as a series of restrictions. The LBS then makes choice between new appropriate storage nodes and existing scheduling scenario by analyzing relations between restrictions so that the reusability of scheduler scenarios can be improved. The LBS maintains the balance between the reusability of policy and the load of storage, and achieves the goal of load balance finally. The simulation and comparisons show that the LBS outperforms the Random and the Least obviously in load balancing, policy decupling and scalability, and that the load balance of the LBS improves 10 times against the two baseline algorithms.

Keywords: application-level storage aggregating; schedule algorithm; load balance

随着信息技术的发展，特别是互联网应用的普及，许多行业产生或处理的数据迅速膨胀，这在数字电视、传感器应用、支持“云计算”的计算中心和社会网络等领域尤为明显. 国际数据公司(IDC)[1]的研究显示，2007年以数字形式创建、捕获或复制的信息数量为281 EB，到2011年这个数字将达到1 800 EB，年增长率可达到60%. 在这种背景下，应用对存储提出了更高的要求，诸如存储容量、数据访问性能、数据传输性能、数据管理能力、存储扩展能力等.

随着云计算的兴起，以Google文件系统、Amazon的S3为代表的应用层虚拟化存储技术迅速发展，这种低成本、高可用的存储方式代表了当前的一种发展趋势. Google文件系统将廉价的存储设备聚合起来，形成一个海量的分布式文件系统，具有良好的伸缩性、可靠性、可用性，可以处理频繁发生的组件故障，也是Google其他应用的基础. S3是Amazon提供的一个在线存储服务，类似于Google文件系统，它通过简单的Web界面，以较低的成本提供

收稿日期：2009-02-11. 作者简介：颜秉珩(1982—)，男，博士生；钱德沛(联系人)，男，教授，博士生导师. 基金项目：国家高技术研究发展计划资助项目(2006AA01A106，2006AA01A124)；科技部国际合作项目(2006DFA11080)；欧盟IST资助项目(IST-2006-045609).

具有伸缩性、高可用、低延迟的存储系统，可以存储任意大小的数据对象（多达 5 GB），数据对象通过 HTTP 协议或者 BT 协议访问.

然而，在很多领域的遗留存储系统仍然是重要的数据来源，应用层存储聚合不能忽视它们. 因此，本文设计了一种支持应用层存储聚合的中间件（αFS），其目标之一就是整合遗留存储系统. 图 1 是 αFS 的架构，其主要组件采用沙漏型架构，以数据资源访问与管理（DRAM）协议为核心，通过配置合适的驱动程序，向下管理和访问各种异构的存储系统，通过基础服务（例如调度和代理服务）向上提供底层存储资源的统一访问和管理界面. 调度服务是 αFS 中的基础服务之一，解决数据存到何处的问题. 为此，本文提出了一种支持负载的存储调度（LBS）算法，将作业调度的思想融入存储调度之中，既借鉴了作业调度的经验[2]，又针对存储调度的特点进行了改造. LBS 算法可以满足数据存储的多样化需求，提高数据的可用性，保证 αFS 的负载均衡.

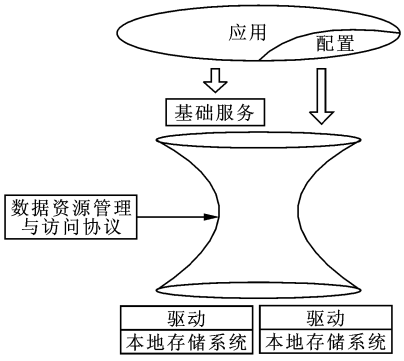

图 1 αFS 的架构

1 存储调度解决的问题

为了提高数据的可用性，αFS 选择了若干存储节点存放文件的副本，当失效副本达到一定数量时，αFS 会重新选择存储节点来创建副本，维持一定的副本量. 如何选择存储节点就是调度模块的任务，对于调度来说，αFS 具有以下特点.

（1）底层存储系统差异巨大. 存储资源在容量、安全、可用性等方面同样存在着差异，调度算法需要充分考虑这些数据才能做出正确的存储方案.

（2）数据的存储需求存在差异. 不同类型数据的存储需求是不一样的，调度算法需要灵活地满足它们的存储要求.

调度模块需要综合考虑存储系统和数据存储的需求，即在保证数据可用性的同时，维持 αFS 底层存储系统的负载均衡. 调度模块无需负责副本的一致性维护和副本数据的传输. 调度模块的核心算法是 LBS，其设计要点包含以下内容.

（1）在满足数据存储需求的前提下，尽量减少节点存储文件的数量，这样可以降低数据失效的风险和重建副本的开销.

（2）复用已有的调度方案，减少策略间的耦合，降低修复副本的复杂度.

（3）只要系统内存符合节点要求，LBS 需保证调度结果可用.

2 LBS 算法

LBS 分为下面 3 个阶段进行.

（1）优先选择新加入的符合条件的存储节点（即自由节点），否则进入第 2 阶段.

（2）考察过去的调度方案，寻找是否存在可以复用的方案.

（3）在系统内寻找可用的存储节点，同时维护系统的负载均衡.

2.1 定义

给定一个存储节点集合 $N=\{n_1,\cdots,n_i,\cdots\}$，每个节点 n_i 的信息可以通过一系列属性/值对描述. 例如，节点 n_i 的容量是 4 GB，支持 FTP 协议等，使用 $F(n_i)$ 表示 n_i 的属性集合，并定义 $F(N)=\bigcup_{n\in N}F(n_i)$，它是所有节点的属性集合，简写为 F.

对于任意的 $n_i\in N$，都可以表示为一个集合 $n_i=\{v_1,\cdots,v_{|F|}\}$，其中 v_i 是节点 n_i 关于 F 中第 i 个属性的值.

定义节点的文件负载为存储在该节点上的文件数目，用 $l_F(n_i)$ 表示.

约束 $\Phi=\{\varphi_1,\cdots,\varphi_{|F|}\}$，$\varphi_i$ 是关于 F 中第 i 个属性的约束函数. 给定一个属性值 v_i，如果 v_i 满足 φ_i，则 $\varphi_i(v_i)=\text{true}$，否则 $\varphi_i(v_i)=\text{false}$. T 是一个永真的约束函数，即 $T=\text{true}$，表示对相应的属性不做任何限制. 对于节点 $n_i=\{v_1,\cdots,v_{|F|}\}$ 满足约束 Φ，当且仅当对于 $\forall\varphi_i\in\Phi$，$\varphi_i(v_i)=\text{true}$ 成立，记作 $\Phi(n_i)=\text{true}$. 那么，N 中满足约束 Φ 的节点集合 $\Phi(N)=\{n_i\mid n_i\in N \text{ and } \Phi(n_i)=\text{true}\}$. 关于两个约束之间的关系，定义如下.

定义 1 对于 $\forall n\in N$，如果 $\varphi(n)=\text{true}\Rightarrow\varphi'(n)=\text{true}$ 成立，则称 $\Phi\subseteqq\Phi'$.

显然有 $\Phi\subseteqq T$ 成立. 虽然借助 n 来定义 2 个约束之间的关系，但这种关系是约束之间的内在关系，与具体的 n 无关. 策略 $P=\langle D,\Phi,k\rangle$ 是一个三元组，

其中 $D \subseteq N$ 是 P 的生成集合，Φ 是 P 的约束，k 是需要满足 Φ 的存储节点数. P 的文件负载是采用该策略存储的文件数，用 $l_F(P)$ 表示.

定义 2　P 是部分有效的，当且仅当 $D_P \subseteq \Phi(N)$.

定义 3　P 是有效的，当且仅当 $D_P \subseteq \Phi(N)$ 和 $|D_P| \geqslant k$ 同时成立.

定义 $N_P = \bigcup_{P \in E_P} D_P$，它是所有策略生成集合的并集，其中 E_P 是已有策略的集合. 如果一个节点不属于任何一个已有策略的生成集合，则称该节点为自由节点. 自由节点的集合 $S = N - N_P$.

节点的策略负载定义为

$$l_P(n_i) = \sum_{P \in E_P} n_i = \begin{cases} 1 & n_i \in D_P \\ 0 & n_i \notin D_P \end{cases}$$

对于一个自由节点，其策略负载为 0.

2.2　策略的创建

根据第 2.1 节的定义，LBS 算法的目标就是获得一个有效的或者部分有效的策略，其生成的集合包含着文件存储的目标节点，这正是一个创建策略的过程.

LBS 算法的第 1 个阶段（算法 1）如图 2 所示，输入是 S、Φ 和 k，输出是一个部分有效的策略. 算法试图仅用 S 来创建一个有效的策略，则输出策略的生成集合应包含满足约束的自由节点集合，所以算法的复杂度为 $O(|S||F|)$. 一般说来，$|F|$ 少于 10 项，因此算法复杂度与自由节点数呈线性关系.

```
1  CREATE_POLICY_Ⅰ(S,Φ,k)
2  begin
3      D←∅;
4      P←⟨D,Φ,k⟩;
5      foreach s∈S do
6          if Φ(s)=true then
7              D←D∪{s};
8              k←k-1;
9          if k≤0 then break;
10     end
11     return P;
12 end
```

图 2　使用自由节点集合创建一个策略（算法 1）

如果返回的策略不是有效的（见定义 3），就会执行 LBS 算法的第 2 阶段（算法 2），如图 3 所示. 算法试图复用一个已有的策略来存储当前的数据，其输入是当前的 E_P、Φ 和 k，输出是一个已有的策略.

算法 2 的核心思想是：选择满足 $\Phi_{P'} \subseteq \Phi$（算法 2 第 6 行）并且对负载较小的策略进行复用；防止策略的过度复用（算法第 5 行），以免影响负载均衡；倾向于选择有效的策略（算法第 8 行～第 10 行），同时保证输出的策略具有与 k 最为接近的生成集合，以形成满足约束的最佳匹配；在生成集合大小相同的情况下，选择负载较小的策略（算法第 11 行～第 13 行）. 算法 2 的复杂度是 $O(|E_P|)$.

```
1  CREATE_POLICY_Ⅱ(E_P,Φ,k)
2  begin
3      P ← null;
4      foreach P′∈E_P do
5          if l_F(P)>threshold then continue;
6          if Φ_P′⊆Φ then
7              if P=null then P ← P′;
8              else if ||D_P|-k|>||D_P′|-k| then
9                  if |D_P|<k or |D_P′|>k then
10                     P ← P′;
11             else if ||D_P|-k|=|D_P′|-k| then
12                 if l_F(P)> l_F(P′) then
13                     P ← P′;
14     end
15     return P;
16 end
```

图 3　复用已有策略（算法 2）

如果没有可以复用的策略，或者返回的策略不是有效的，LBS 算法将执行第 3 阶段（算法 3），如图 4 所示，输入自由节点集合 S、非自由节点集合 N_P、约束 Φ 和所需节点数目 k，输出一个有效的策略.

由图 4 知，如果存在符合约束的自由节点，算法终止，否则在系统中会寻找满足约束的存储节点，并将它们合并起来形成一个有效的策略. 算法 3 主要反映算法的复杂度，由于它采用了快速排序（算法第 5 行），所以复杂度约为 $O(|N_P|\lg|N_P|)$，又因循环（第 6 行～第 10 行）的复杂度为 $O(|N_P||F|)$，通常 $|F|$ 较小，所以复杂度与 $O(|N_P|)$ 呈线性关系.

```
1  CREATE_POLICY_Ⅲ(S,N_P,Φ,k)
2  begin
3      P←CREATE_POLICY_Ⅰ(S,Φ,k);
4      if |D_P|≥k then return P;
5      sort N_P by l_F(n∈N_P) ascendly;
6      foreach n_i ∈N_P do
7          if Φ(n_i)=true then
8              D_P←D_P∪{n}
9          if |D_P|≥k then return P;
10     end
11     return null;
12 end
```

图 4　遍历节点集合创建一个策略（算法 3）

2.3　策略的修复

当 D_P 中的失效节点数超过设定的阈值时，就需要重建一些副本，而保持一定的副本数量，是一个修复策略的过程（算法 4），如图 5 所示，输入是 S、N_P、E_P 及失效节点 n_f. 使用前面的策略创建算法可选择合适的存储节点来重建副本，在最坏的情况下它的复杂度约为 $O(|E_P||N|)$.

```
1  RECOVER_POLICY(S, N_P, E_P, n_f)
2  begin
3      foreach policy P whose D_P contains n_f do
4          D_P ← D_P − {n_f};
5          l = k_P − |D_P|;
6          if l < threshold then continue;
7          create policy Q with (S, N_P, Φ_P, l);
8          foreach n_i ∈ D_Q do
9              create replica on node n_i;
10         end
11         D_P ← D_P ∪ D_Q;
12     end
13 end
```

图 5 策略的修复算法(算法 4)

3 模拟与结果分析

为了验证 LBS 算法的性能,本文设计了模拟实验,并将 LBS 与其他算法进行比较,结果表明 LBS 的性能明显优于其他简单的调度算法.

3.1 模拟环境

使用 SimJava[3] 来构建模拟环境. 实验中共有 3 种事件:文件存储事件、节点加入事件和节点失效事件,其中文件存储事件同时产生该文件对目标存储节点的约束. 假定存储节点之间是互相独立的,并具有相似的失效模型,其有效工作时间满足指数分布.

节点在时间 τ 内的失效概率为 $1-e^{-\tau/\lambda}$,其中 λ 是节点正常工作时间的平均值,λ 较大就意味着节点的可靠性较高,实验中 λ 的平均值为 5 a. 文件存储事件满足泊松分布,即单位时间(1 d)内产生 k 个文件存储事件的概率为 $\frac{\mu^k}{k!}e^{-\mu}$,实验中设定 μ 的平均值为 100. 节点加入事件与文件存储事件满足同样的概率分布,实验设定每月平均加入 1 个新节点.

在初始环境为 1 000 个存储节点下,模拟了系统在 10 a 中的演化情况. 文献[4]指出,3 个副本已足以保证数据的可用性. 为了充分保证数据的可用性,我们设定 αFS 中的副本数为 4. 在实验的过程中,将策略文件负载的阈值(图 3 第 5 行)设定为最大策略文件负载的 0.8 倍,这是一个经验数值,阈值越大,策略复用越多,反之则策略复用较少. 0.8 倍的设定在模拟环境中表现出了最佳的策略复用和负载的均衡性.

实验中共比较了 3 种算法,即 LBS、Random 和 Least 算法. LBS 算法如前所述,Random 算法在满足约束的节点集合中随机选取节点,Least 在满足约束的节点集合中选择最久最少使用的节点.

3.2 指标

为了度量负载的均衡性和策略的耦合性,实验使用如下 3 个指标[5],每个指标均为 100 次模拟的平均值.

(1)节点均衡度 $B(N_P)=\frac{\max\{l_P(n_i)\}}{\min\{l_P(n_i)\}}$,其中 $n_i\in N_P$、$B(N_P)$反映了全局范围内存储节点使用的均衡情况,但未考虑节点之间的差异.

(2)策略均衡度 $B(P)=\frac{\max\{l_F(n_i)\}}{\min\{l_F(n_i)\}}$,其中 $n_i\in D_P$、$B(P)$反映了一个策略内节点使用的均衡情况,因此兼顾了节点差异. 为了度量系统的策略均衡性,应使用 $B(P)$的平均值,即

$$B(E_P)=\frac{1}{|E_P|}\sum_{P\in E_P}B(P)$$

(3)节点策略数 $V_P=\frac{|E_P|}{|N|}$,V_P 反映了策略间的耦合情况,数值越大,策略耦合性越强.

3.3 模拟结果

图 6 显示了 10 a 内节点均衡性的演化情况,可以看出 Least 和 Random 算法的结果差别不大,而 LBS 的节点均衡性更优,$B(N)$指标提升了 10 倍左右,这与 LBS 的策略复用是分不开的. 图 7 显示了策略均衡性在 10 a 内的演化情况,Least 算法的策略均衡度优于 Random 算法,但不如 LBS 算法. LBS 的策略构建方法与策略复用在此起到了重要的作用,将系统的策略均衡性提高了约 1.5 倍左右.

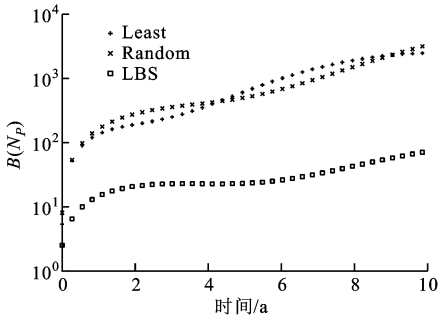

图 6 $B(N_P)$在 10 a 内的演化情况

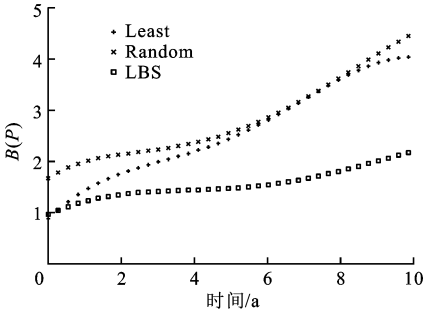

图 7 $B(P)$在 10 a 内的演化情况

综合以上 2 个指标可以看出，LBS 在节点均衡性、策略均衡性上明显优于简单的调度算法，从整体上提高了系统的负载均衡.

图 8 显示了节点策略数在 10 a 内的演化情况，由于 Least、Random 算法没有策略复用，因此它们的节点策略数比较接近，而 LBS 的策略复用效果明显，大大降低了生成策略的数目，其策略数仅为普通情况下的 1%.

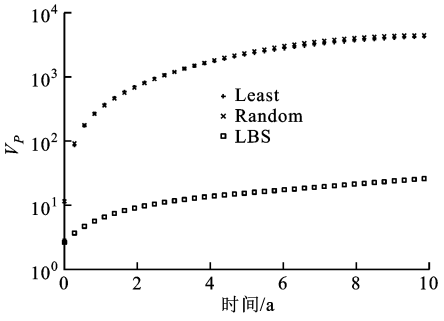

图 8　V_P 在 10 a 内的演化情况

4　结束语

本文给出的 LBS 是一种集中式控制环境下的调度算法，它可以充分发挥存储节点的性能，按需分配存储资源，达到负载均衡的目的. 模拟结果表明，与基准算法相比，LBS 的负载均衡性指标均有显著的提升.

参考文献：

[1] GANTZ J. The diverse and exploding digital universe: an updated forecast of worldwide information growth through 2011 [M]. Framingham, MA, USA: IDC, 2008.

[2] 胡周君，胡志刚，李林. 一种基于性能评估的元任务调度算法 [J]. 西安交通大学学报，2008，42(8)：972-976.
HU Zhoujun, HU Zhigang, LI Lin. Performance evaluation based meta-task scheduling algorithm [J]. Journal of Xi′an Jiaotong University, 2008, 42(8): 972-976.

[3] MCNAB R, HOWELL F. Using java for discrete event simulation [C]// Proceedings of 12th UK Computer and Telecommunications Performance Engineering Workshop. Edinburgh, UK: UKPEW, 1996: 219-228.

[4] GABBER E, FELLIN J, FLASTER M, et al. Starfish: highly-available block storage [C]// Proceedings of the Freenix Track: 2003 USENIX Annual Technical Conference. San Antonio, Texas, USA: USENIX, 2003: 151-163.

[5] CRAINICEANU A, LINGA P, MACHANAVAJJHALA A, et al. P-ring: an efficient and robust P2P range index structure [C]// Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. New York, USA: ACM, 2007: 223-234.

（编辑　苗凌）

两项教育部科技创新工程重大培育项目通过验收

2009 年 7 月 14 日，西安交通大学万明习教授承担的教育部重大培育资金项目“肿瘤热消融瞬态物理和超声功能与间接分子监控成像”和藏伟进教授承担的“副交感神经对正常及疾病状况下心血管调控的机制研究”验收会在西安交通大学召开，教育部科技司计划处李渝红处长出席会议并发表了重要讲话. 会议邀请全国同行业知名专家对项目进行了评议，专家们给予项目高度评价，并一致同意通过验收，西安交通大学科研院贾毅华院长出席了会议.

验收会上，验收专家组成员认真审核了项目的任务书、结题报告、研究报告和有关的技术资料，听取了项目负责人的汇报，并进行了实地考察，与会专家一致认为：项目研究内容设置合理，技术路线正确，方法科学，具有创新性和临床应用潜力，成果显著. 专家组成员对课题组认真、踏实的科研态度和学风给与了高序赞扬.

这两项科技创新工程重大培育资金项目立项于 2005 年，共获得资助经费 70 万元，在课题组的努力和学校的支持下，经过三年的研究取得了一系列重要成果，培养了一大批优秀人才，获得了国家自然科学基金重点项目、973 项目课题等资助，实现了项目的培育目的，为进一步深入研究提供了坚实基础.