云存储系统管理节点故障自恢复算法①

马玮骏[1], 王　强[1], 何晓晖[1], 张　舒[2], 张　庆[3]

[1](解放军理工大学 野战工程学院, 南京 210014)
[2](西部战区空军气象中心, 成都 610000)
[3](东部战区空军气象中心, 南京 210018)

摘　要: 为了解决大规模云存储系统中管理节点发生故障导致存储服务不可用的问题, 建立了管理节点故障影响分析模型, 提出了一种基于消息的管理节点动态自我恢复算法 FRA-M. 该算法通过基于负载均衡的元数据备份更新控制方法完成多个管理节点之间相互协作、透明接管和故障自我恢复. 测试结果表明, FRA-M 算法能够使得管理节点发生故障时自动进行切换, 并且能够合理地分配资源达到良好的负载均衡状态. 通过控制 TCP 超时时限、故障检测周期以及故障检测超时, 能够使得 FRA-M 算法的性能保持在相对稳定的区间, 随失效时刻的适应性也比较强. 当管理节点发生故障时, FRA-M 能够较好地保证存储服务可用性、数据可用性和数据可靠性.

关键词: 云存储系统; 管理节点; 自我恢复; 元数据; 负载均衡; 动态切换

Fault Self-Recovery Algorithm for Management Node in Cloud Storage System

MA Wei-Jun[1],WANG Qiang[1], HE Xiao-Hui[1], ZHANG Shu[2], ZHANG Qing[3]

[1](Institute of Field Engineering, PLA University of Science and Technology, Nanjing 210014, China)
[2](Western Theater Air Meteorological Center, Chengdu 610000, China)
[3](Eastern Theater Air Meteorological Center, Nanjing 210018, China)

Abstract: In order to solve the storage service unavailable problem on account of the management node fault in huge cloud storage system, an analysis model for fault effect of management node is built and a dynamic self-recovery algorithm for management node based on message called FRA-M is presented. FRA-M implements the cooperation, transparent take-over and self-recovery of management nodes by metadata update control based on load balance. Experiment shows FRA-M can provide management nodes auto switching when fault occurs and achieve good load balance by favorable resource allocation. The performance of FRA-M is also maintained in a relatively stable interval by reasonable control of TCP timeout, fault detection cycle and fault detection timeout. The storage service availability, data usability and data reliability are guaranteed by FRA-M during the breakdown of management nodes.

Key words: cloud storage system; management node; self-recovery; metadata; load balancing; dynamic switching

大规模云存储系统(Huge Cloud Storage System, 以下简称 HCSS)由于其规模大、覆盖范围广、访问量大、存储数据量大等特点, 逐步成为业界的研究热点[1], 而故障恢复问题一直是 HCSS 的重要研究内容与方向. 传统的云存储系统往往都通过人工或者半自动化的方式实现故障的恢复, 效率低下并且容易出现误操作而导致一系列的故障连锁反应, 在规模较大的 HCSS 中更是举步维艰. 因此, 实现故障的自我恢复是 HCSS 中需要解决的关键问题之一.

文献[1]给出了一种提高云存储系统可靠性的方法, 并提出了多个云存储管理节点相互协作提供存储服务高可用和高可靠性的概念. CMU 开发的 Coda[3], 采用服务器复制技术将文件副本存储在多个服务器上来保证网络故障时的数据可用性和数据可靠性. Globus 中包含的 GASS[4,5], 提供了广域网范围内的数据移动与访问服务, 该系统为每个节点提供多条存储路径, 结

① 基金项目:国家自然科学基金(61371119)
收稿时间:2016-05-10;收到修改稿时间:2016-06-30　[doi:10.15888/j.cnki.csa.005568]

合副本的使用, 以保证数据可用性和可靠性. UC Berkeley 基于 P2P 思想设计的支持广域网范围海量文件存储的 OceanStore[6], 为减小空间和带宽消耗使用纠删码存储归档的数据, 同时使用完整副本来提高数据访问的效率. MIT 的 CFS[7], 采用复制及缓存的方法, 对大文件实施分块复制, 对小文件进行缓存, 以保证可靠性和负载均衡. 基于云计算技术的海量云存储系统 GFS[8], 采用副本技术和 Master 节点备份技术来保证访问性能和可靠性. 文献[9]从资源分配角度建立了云存储系统可靠性分析模型, 并对模型的有效性进行了验证. 文献[10]基于云存储节点数据访问频率, 提出一种副本备份算法, 提高了数据的可靠性和云存储访问性能. 为了提高云存储服务的可用性和容灾性能, 近年来一些研究人员开始使用 P2P 技术融合至云存储架构中[11-13].

可以看出, 各类云存储系统必须能够通过有效的软硬件动态恢复技术使得系统在发生故障时能够自动恢复软件、硬件的正常运行, 尽可能地提供持续、正确的存储服务, 保证系统存储服务的高可用性, 这样才能保证数据的高可用性.

本文围绕 HCSS 中管理节点故障失效时如何持续提供云存储服务的问题展开研究, 给出 HCSS 管理节点故障分析模型, 并以该模型为基础提出 HCSS 管理节点故障自我恢复的相关算法, 最后进行实验和分析证明该算法的有效性.

1 管理节点故障分析模型

HCSS 管理节点主要负责为云存储客户端提供元数据服务, 同时需要检测各个存储节点的工作状态, 通常 HCSS 中有多个管理节点, 其中有一个作为主节点, 负责所有管理节点的故障检测. 使用故障管理对象、故障的限制条件和表现以及故障管理对象的状态三个维度来表示 HCSS 中管理节点的故障分析模型.

设 HCSS 管理节点的故障状态空间 $S_{CSSM}=\{s_1,s_2,s_3,...,s_n\}$; HCSS 管理节点故障的限制条件和表现空间 $C_{CSSM}=\{c_1,c_2,c_3,...,c_m\}$; HCSS 管理节点实时状态空间 $R_{CSSM}=\{r_1,r_2,r_3,...,r_l\}$; 在 t 时刻, 存储服务可用性为 $SA(t)$; 数据可靠性为 $DR(t)$; 数据可用性为 $AD(t)$.

记故障发生时间为 t_1, 故障状态为 s_m, HCSS 管理节点为 o_i, HCSS 管理节点实时状态为 r_j, HCSS 管理节点故障的限制条件和表现为 c_k, 故障持续时间为 Δt, 则故障 s_m 对存储服务可用性的影响记为:

$$F_{s_m\to SA}=f_{s_m\to SA}(o_i,r_j,c_k)\in[0,1] \quad (1)$$

设 $t_2>t_1$, $t_2\in(t_1,t_1+\Delta t]$, 则 t_2 时刻系统存储服务可用性记为:

$$SA(t_2)=SA(t_1)(1-F_{s_m\to SA}) \quad (2)$$

式(1)中 $f_{s_m\to SA}$ 表示故障 s_m 对存储服务可用性的影响函数, 该函数的取值范围为闭区间[0,1], 取 0 时表示没有影响, 取 1 时表示影响最大, 使得系统存储服务可用性为 0, 即无法提供存储服务.

同理, 设 $f_{s_m\to DR}$ 和 $f_{s_m\to AD}$ 分别表示故障 s_m 对系统数据可靠性和系统数据可用性的影响函数, 则 t_2 时刻系统数据可靠性记为:

$$DR(t_2)=DR(t_1)(1-F_{s_m\to DR}) \quad (3)$$

t_2 时刻系统数据可用性记为:

$$AD(t_2)=AD(t_1)(1-F_{s_m\to AD}) \quad (4)$$

为了讨论简便, 下面对影响函数进行两值化:

$$F_{s_m\to SA}=f_{s_m\to SA}(o_i,r_j,c_k)\in\{0,1\} \quad (5)$$

$$F_{s_m\to DR}=f_{s_m\to DR}(o_i,r_j,c_k)\in\{0,1\} \quad (6)$$

$$F_{s_m\to AD}=f_{s_m\to AD}(o_i,r_j,c_k)\in\{0,1\} \quad (7)$$

式(5)~(7)表示将故障影响简化为 0 和 1 两种取值, 0 表示没有影响, 1 表示有影响, 这种简化对于界定故障的检测范围、恢复范围以及恢复时效是很有意义的, 对于故障影响为 0 的故障, 可以检测但不一定要及时恢复, 对于故障影响为 1 的故障, 是必须要进行检测和及时恢复的.

下面给出 HCSS 管理节点在崩溃故障情况下的故障分析如表 1 所示. 这里的崩溃故障指节点宕机、掉电或者系统崩溃等故障, 对 HCSS 管理节点上存储的元数据随节点崩溃的丢失情况不做任何假设.

表 1 HCSS 管理节点崩溃故障影响分析

r_j	$F_{s_m\to SA}$	$F_{s_m\to DR}$	$F_{s_m\to AD}$
r_1=空闲状态	1	0	1
r_2=响应客户端读请求	1	0	1
r_3=响应客户端写请求	1	1	1
r_4=云存储节点检测	1	1	1
r_5=管理节点检测	1	1	1
r_6=元数据同步	1	1	1
r_7=响应检测请求	1	0	1

r_1: 云存储客户端访问时, 会造成无法服务, 影响存储服务可用性以及数据可用性.

r_2: HCSS 管理节点无法继续提供云存储服务, 影响存储服务可用性以及数据可用性.

r_3: HCSS 管理节点无法继续提供云存储服务, 影响存储服务可用性以及数据可用性, 由于是写数据, 数据可靠性也受到影响.

r_4: 隶属于该 HCSS 管理节点检测范围的云存储节点状态将无法更新, 其他 HCSS 管理节点只能使用旧的云存储节点状态信息进行存储任务管理, 一旦有云存储客户端访问, 将影响存储服务可用性以及数据可用性. 由于云存储节点是存储用户数据的实体, 其状态正常与否将直接影响系统的数据可靠性.

r_5: 首先影响的是系统的存储服务可用性以及数据可用性, 由于此时 HCSS 管理节点为状态、故障检测主节点, 该节点崩溃会导致其他 HCSS 管理节点无法获取全局的状态信息, 这必然会影响元数据管理工作, 因此系统数据可靠性也会受到影响.

r_6: 首先影响的是系统的存储服务可用性以及数据可用性, 如果正在向别的 HCSS 管理节点发送元数据, 则无法保证元数据的完整性和一致性, 从而会影响系统的数据可靠性.

r_7: 首先影响的是系统的存储服务可用性以及数据可用性, 由于元数据和云存储节点上的数据都没有发生变化, 因此系统数据可靠性不会受到影响.

通过对 HCSS 管理节点的故障影响情况分析可以看出, 对于管理型节点的自我恢复, 关键在于故障恢复软件能否在 HCSS 管理节点发生故障时, 在各种实时状态下, 使多个 HCSS 管理节点之间通过相互备份的机制实现透明接管, 否则某个 HCSS 管理节点故障将导致系统的存储服务可用性、数据可用性都受到影响, 也会影响系统的数据可靠性.

2 基于消息的管理节点自恢复算法FRA-M

从故障影响分析模型的分析结果可以看出, 实现HCSS管理节点故障动态切换, 一方面需要从HCSS管理节点发生故障之前的处理机制考虑, 保证发生故障时, 元数据不会丢失(即便正在更新元数据), 从而不影响系统的存储服务可用性、数据可用性以及数据可靠性, 重点就在于更新元数据时的控制算法; 另一方面则需要从其他 HCSS 管理节点的接管机制考虑. 本文提出一种基于消息的管理节点动态自我恢复算法FRA-M(Fault self-Recovery Algorithm for Management node based on message), 从元数据更新和云存储节点管理分配角度研究了管理节点故障自我恢复的相关方法.

2.1 管理节点故障自我恢复元数据更新算法

管理节点故障自我恢复元数据更新算法伪代码如下所示:

```
1:  ON RequestFromCSSAN(d_i) then
2:  if NeedUpdateMetaData then
3:  begin
4:      UpdateMetaData();
5:      S_CSSMN = S_CSSMN - h_j;
6:      Select random h_i in S_CSSMN
7:      if SendMeta(d_i, h_i ,meta,ID)=OK then
8:      if SendResponse(meta, h_i ,ID)=OK then
        SendConfirm(h_i) ;
9:      else S_CSSMN ← S_CSSMN − h_i ; goto 6;
10: end
11: else SendResponse();
12: ON ReceiveFromBackMeta(ID,MetaData, h_j) then
13:     TempSaveMetaData(ID,MetaData);
14:     ComputeTimeout(MetaData,T);
15:     SetTimer();
16: ON ReceiveBackMetaDataConfirm(ID, h_j) or
    ReceiveReport(ID, d_i) then
17:     SaveMetaData(MetaData);
18:     CloseTimer();
19: ON Timer>T then DeleteMeta(MetaData);
```

元数据更新算法的核心思想在于: HCSS管理节点需要更新元数据时, 必须找到一个能够暂时备份元数据的节点进行元数据备份, 然后再响应云存储客户端的请求, 这样做的优点在于无论何时 HCSS 管理节点崩溃, 系统都可以自动恢复, 分析如下:

① 当 h_j 在响应 d_i 请求前崩溃, 则 d_i 可以按照算法 3.1 自动进行恢复;

② 当 h_j 在响应 d_i 请求后并且发送了确认信息给 h_i 后崩溃, 元数据仍然备份于其他 CSS 管理节点, 并且可以得到同步, 数据可靠性和可用性都有保证;

③ 当 h_j 在响应 d_i 请求后但没有发送确认信息给 h_i 时崩溃, 由于响应 d_i 请求的信息中包含了 h_i, d_i 完成操作后会汇报完成状态给 h_j 以及 h_i, 这种没有请求的汇报信息对于 h_i 来说是一个确认信息, 用于让元数据

生效. 这样便不会产生孤儿数据的问题, 保证了系统的数据可用性和数据可靠性;

④ 当 h_j 在响应 d_i 请求过程中崩溃, d_i 可以按照算法 3.1 自动进行恢复; 而 h_i 无法得到确认信息, 过一段时间之后会将元数据删除, 因此不会存在元数据失效问题.

2.2 管理节点故障自我恢复动态接管算法

元数据更新算法保证了 HCSS 管理节点在状态 r_3、r_6 下发生故障时系统的数据可靠性, HCSS 管理节点还有一项重要的功能便是检测云存储节点的实时状态以及故障, 即状态 r_4, 由于每个 HCSS 管理节点都需要检测一部分云存储节点, 因此当发生某个 HCSS 管理节点发生故障时必须能够由其他 HCSS 管理节点接管该任务. 动态接管算法伪代码如下所示:

```
1:  ON ReceiveCSSMNStatus() then
2:  if  hj.Status=ERROR  then;
3:  begin
4:      RCSSCOUNT ← total node in S_CSSMN ;
5:      LC←hj.SNODE.Count;
6:      SN←Sequence Number of hi in S_CSSMN;
7:      LA = ⌈ LC / RCSSCOUNT ⌉ ;
8:      if LA<=1 and SN< LC  then
        hi.SNODE←hi.SNODE ∪ hj.SNODE[SN];
9:      else if LA>1 and NS*LA<= hj.SNODE.Count
        then  hi.SNODE ← hi.SNODE ∪ Y_{k=(SN−1)×LA+1}^{SN×LA} hj.SNODE[k]
10:     else if LA>1 and NS*LA> hj.SNODE.Count
        and (NS-1)*LA< hj.SNODE.Count
        then hi.SNODE ← hi.SNODE ∪ Y_{k=(SN−1)×LA+1}^{LC} hj.SNODE[k]
11: end;
```

动态接管算法假设当前执行接管任务的 HCSS 管理节点为 h_i, 该算法主要针对 HCSS 管理节点在状态 r_4 时发生故障的动态恢复处理, 其中第 1~2 行表示收到 HCSS 管理故障检测主节点的检测结果, 则判断是否有 HCSS 管理节点出现故障, 如果 HCSS 管理节点 h_j 出现故障, 则进行 h_j 的云存储节点检测工作接管, 第 4 行中 *RCSSCOUNT* 表示 S_{CSSMN} 中正常节点的数目, 第 5 行中 *LC* 表示 h_j 所负责检测的云存储节点数目, 第 6 行中 *SN* 表示节点 h_i 在 S_{CSSMN} 正常节点中的顺序(有节点发生故障, 节点顺序可能发生变化), 第 7 行 *LA* 表示每个 S_{CSSMN} 正常节点接管的云存储节点数目, 第 8~10 行表示将 h_j 原来负责检测的云存储节点在 S_{CSSMN} 正常节点上进行均分的算法, 这里主要计算节点 h_i 所分配到的云存储节点, 分为 3 种情况:

① HCSS 管理节点数目大于需要分配的云存储节点数目, 则按顺序进行分配, 每个 CSS 管理节点分配一个;

② HCSS 管理节点数目小于需要分配的云存储节点数目, 则每个 HCSS 管理节点按照顺序分配 LA 个;

③ 如果按照 LA 的分配方式不够分配, 则按序分配, 分配完为止.

另外一个需要注意的问题是该算法中 *SN* 的计算方式由故障检测主节点指定, 可以正序可以倒序, 指定方式为正序和倒序穿插进行, 这样的优点在于不会使得某些排序靠后的 HCSS 管理节点每次都不能分配到云存储节点.

动态接管算法的优点在于各个正常的 HCSS 管理节点能够自行计算所需接管的云存储节点, 并且实时进行接管, 同时能够保证接管的全局一致性, 等到下一个故障检测周期, 各个 HCSS 管理节点都能够向故障检测主节点提交检测结果, 全局结果能够发布至各个 HCSS 管理节点. 因此, 动态接管算法在 HCSS 管理节点处于状态 r_4 时出现故障的情况下, 保证了系统的数据可靠性.

3 实验与分析

实验中采用 5 个 HCSS 管理节点, 20 个云存储节点, 1 个云存储客户端. 管理节点自恢复测试主要针对 FRA-M 算法, 为了体现测试的全面性, 从云存储客户端和 HCSS 管理节点两个角度对算法进行测试, 一方面测试云存储客户端在 HCSS 管理节点失效后的工作状态以及性能; 另一方面测试某个 HCSS 管理节点失效后其他 HCSS 管理节点的工作状态, 包括故障恢复时间、存储节点分配等.

测试的初始设置如下所示:

① 云存储客户端: TCP 超时时限为 1000ms, 测试次数为 100, 每次测试的请求数为 200, 初始目标 CSS 管理节点编号为 1, 初始故障检测主节点编号为 5;

② HCSS 管理节点: 故障检测超时初始值为 800ms 和 200ms, 故障检测周期为 1000ms, 初始故障

检测主节点编号为 5, 故障检测主节点申请回执超时 MDT 为 100ms, 测试次数为 100.

3.1 云存储客户端测试与分析

云存储客户端主要测试当 HCSS 管理节点失效时, HCSS 代理节点的工作状态, 假设 HCSS 代理节点随机请求 HCSS 管理节点, 记录请求响应时间.

测试过程中, 5 个 HCSS 管理节点在云存储客户端完成所有 200 次请求的过程中, 按照 1~4 的顺序停止响应任何请求(模拟崩溃故障, 即失效), 4 个节点全部失效认为是一次测试, 总共测试 100 次, 节点失效的时刻是随机的, 考察 CSS 代理节点当 CSS 管理节点失效时的请求响应时间受故障的影响以及请求平均响应时间随测试次数的关系如图 1 和图 2 所示.

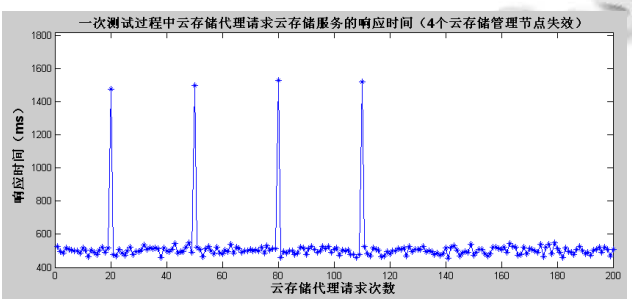

图 1　云存储客户端请求响应时间(4 个节点失效)

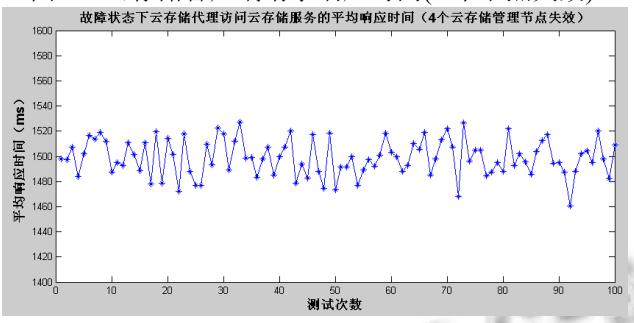

图 2　云存储客户端请求平均响应时间(4 个节点失效)

从图 1 的测试结果可以看出: 当部分 HCSS 管理节点出现故障时(只要不是全部), 云存储客户端仍然可以在有限的时间内获得响应, 说明云存储客户端能够实现自动切换; 云存储客户端大部分请求响应时间都在 500ms 左右, 这是正常情况下云存储客户端请求获得响应的时间, 包括网络时延以及 CSS 管理节点的处理时延. 图 1 中 4 次响应时间较大的点对应 4 个 HCSS 管理节点失效时的情况, 可以看出每次故障时的响应时间都在1500ms左右, 这主要是因为云存储客户端发起请求时, 一旦发现 TCP 连接超时, 便会再次从本地的 HCSS 管理节点列表中取出另外一个可用的 HCSS 管理节点并发起请求, 然后获得响应, 因此总的响应时间应该是 TCP 连接超时的时间(1000ms)与正常请求获得响应的时间之和.

因此总的来说通过控制 TCP 超时时限, 可以使得当 HCSS 管理节点失效时云存储服务故障自恢复的响应时间相对收敛在某个区间之内.

3.2 管理节点测试与分析

HCSS 管理节点测试主要针对当 HCSS 管理节点失效时, 其他 HCSS 管理节点对云存储节点的接管性能和负载均衡情况. 测试过程中, 首先在 HCSS 管理节点 1~3 中按照(1)、(1, 2)、(1, 2, 3)三种组合的顺序停止响应任何请求(模拟崩溃故障, 即失效), 分别表示同时损坏的节点数为 1、2、3, 节点失效的时刻是随机的, 记录节点失效的时刻 t_1 与剩余 HCSS 管理节点更新所管理的云存储节点列表的时刻 t_2 之差 $\Delta t = t_2 - t_1$(接管时间), 用于衡量 HCSS 管理节点失效状态下被接管的性能, 并且输出云存储节点列表(负载均衡结果). 设定故障检测超时初始值分别为 800ms、200ms 进行测试. 测试结果如图 3 和表 1 所示.

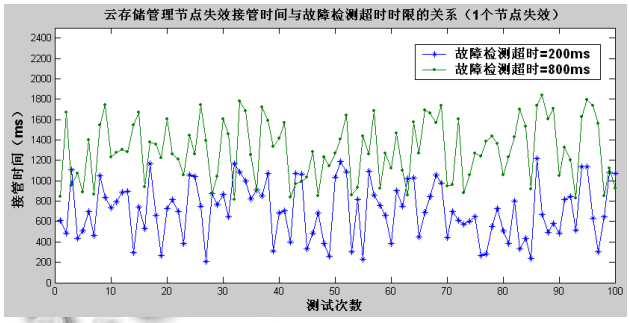

图 3　接管时间与故障检测超时时限的关系

表 1　HCSS 管理节点失效接管负载均衡情况

节点 1	节点 2	节点 3	节点 4	节点 5
1-4	5-8	9-12	13-16	17-20
无	1,5-8	2,9-12	3,13-16	4,17-20
无	无	1-3,9-12	4-6,13-16	7,8,17-20
无	无	无	1-6,13-16	7-12,17-20

图 3 表示 1 个节点失效时, 在不同的故障检测超时设置下的接管时间, 测试结果表明, 当故障检测超时等于200ms时, 接管时间基本上在200ms到1200ms内随机变化, 而故障检测超时等于 800ms 时, 接管时间基本上在 800ms 到 1800ms 内随机变化.

测试表明, 虽然受网络往返时延和节点的计算时

延有很小的影响, 仍然可以通过控制故障检测周期以及故障检测超时, 使得FRA-M算法中接管算法的性能保持在相对稳定的区间, 随失效时刻的适应性也比较强.

表 1 显示了在失效 HCSS 管理节点数不同情况下, 剩余 HCSS 管理节点接管失效 HCSS 管理节点负责检测的云存储节点的情况, 从测试结果可以看出, 初始状态下, 负载是均衡的, 无论失效几个节点, FRA-M 算法中的接管算法都能够使剩余的 HCSS 管理节点很好地进行接管, 并以最平均的状态分配可用 HCSS 管理节点资源, 达到了较好的负载均衡的作用.

4 结语

故障自恢复是大规模云存储系统(HCSS)需要解决的关键问题之一. 本文针对传统的分布式和云存储系统故障恢复效率低下的问题, 建立了 HCSS 中管理节点故障影响分析模型, 基于该模型的分析结果提出了一种基于消息的管理节点动态自我恢复算法 FRA-M, 当 HCSS 管理节点发生故障时, 能够在各种实时状态下, 使多个 HCSS 管理节点之间通过相互备份的方式实现透明接管, 以保证存储服务可用性、数据可用性和数据可靠性.

测试结果表明, FRA-M 算法能够在 HCSS 管理节点发生故障时自动进行云存储服务的恢复, 并且能够合理地分配资源达到良好的负载均衡状态. 通过控制 TCP 超时时限、故障检测周期以及故障检测超时, 能够使得FRA-M算法的性能保持在相对稳定的区间, 随失效时刻的适应性也比较强.

参考文献

1 刘鹏.云计算.北京:电子工业出版社,2015:1–15.

2 马玮骏,吴海佳,刘鹏.MassCloud 云存储系统构架及可靠性机制.河海大学学报:自然科学版,2011,39(3):348–352.

3 Satyanarayanan M, Kistler JJ, Kumar P. Coda: A highly available file system for a distributed workstation environment. IEEE Trans. on Computers, 1990, 39(4): 447–459.

4 Foster I, Kesselman C, Tedesco J. GASS: A data movement and access service for wide area computing systems. Proc. of the Sixth workshop on I/O in Parallel and Distributed Systems. Atlanta, ACM Press. 1999. 78–88.

5 Foster I, Kesselman C. Globus: A metacomputing infrastructure toolkit. International Journal of Supercomputer Applications, 1997, 11(2): 115–128.

6 Kubiatowicz J, Bindel D, Chen Y. OceanStore: An architecture for global-scale persistent storage. Proc. of the ninth international conference on Architectural support for programming languages and operating systems (ASPLOS-IX). New York. ACM Press. 2000. 190–201.

7 Dabek F, Kaashoek MF, Karger D. Wide-area cooperative storage with CFS. Proc. of the eighteenth ACM symposium on Operating systems principles (SOSP'01). New York. ACM Press. 2001. 202–215.

8 Ghemawat S, Gobioff H, and Leung ST. The google file system. Proc. of 19th ACM Symposium on Operating Systems Principles. New York. ACM Press. 2003. 20–43.

9 Faragardi HR, Shojaee R, Tabani H, et al. An analytical model to evaluate reliability of cloud computing systems in the presence of QoS requirements. International Conference on Computer and Information Science. Niigata. ACM Press. 2013. 315–321.

10 Joolahluk J, Wen YF. Reliable and available data replication planning for cloud storage. IEEE International Conference on Advanced Information Networking and Applications. Barcelona. ACM Press. 2013. 772–779.

11 Song J, Deng JH. NOVA: A P2P-Cloud VoD system for iptv with collaborative pre-deployment module based on recommendation scheme. International Conference on Materials Science and Information Technology. Nanjing. Springer Press. 2013. 1566–1570.

12 Jacob C. Cost and profit driven Cloud-P2P interaction. Peer-to-Peer Networking and Applications, 2015, 8(2): 244–259.

13 Babaolu O, Marzolla M, Trambrini M. Design and implementation of a P2P cloud system. Proc. of the ACM Symposium on Applied Computing. Trento. Springer Press. 2012. 412–417.