

## 基于校验编码备份的分布存储方案

陈冬晓<sup>1,2</sup>, 王 鹏<sup>1,3\*</sup>

(1. 中国科学院 成都计算机应用研究所, 成都 610041; 2. 中国科学院研究生院, 北京 100049;

3. 成都信息工程学院 并行计算实验室, 成都 610225)

(\* 通信作者电子邮箱 wp002005@163.com)

**摘 要:**传统的云计算存储系统为保障可用性,一般使用镜像冗余备份而产生大量冗余备份数据,影响了存储数据空间的利用效率。针对此情况,为减少备份数据对存储空间的占用,提出一种存储方案。放弃了镜像冗余备份,引入校验编码的方式进行备份,以减少备份数据;同时采用了冲突跳转的机制对备份进行验证,在保证备份数据有效性的前提下减少备份数量。通过模拟程序运行结果与主流云存储方案的对比表明,所提存储方案在保证数据可靠性的同时,显著地降低了分布存储对磁盘空间的占用。

**关键词:**云存储;一致性哈希;Hadoop 分布式文件系统;数据备份;数据恢复

**中图分类号:** TP301.6 **文献标志码:** A

### Distributed storage solution based on parity coding

CHEN Dongxiao<sup>1,2</sup>, WANG Peng<sup>1,3\*</sup>

(1. Chengdu Institute of Computer Applications, Chinese Academy of Sciences, Chengdu Sichuan 610041, China;

2. Graduate University of Chinese Academy of Sciences, Beijing 100190, China;

3. Parallel Computing Laboratory, Chengdu University of Information Technology, Chengdu Sichuan 610025, China)

**Abstract:** To guarantee reliability, traditional cloud storage solutions generally backup data through mirror redundancy, which influences the usage efficiency of storage data space. A storage solution was proposed to reduce the usage of storage data space for redundancy-backup data. The solution introduced: 1) the parity coding backup instead of mirror backup, which reduced the size of backup data; 2) the conflict-jump mechanism to confirm the backup data, which guaranteed reliability while number of backup data copies was reduced. The contrast between running result of simulation program and performance of mainstream cloud storage solutions shows that, by using the proposed solution, the usage of storage space for distributed storage is significantly reduced while the reliability gets guaranteed.

**Key words:** cloud storage; consistent hash; Hadoop Distributed File System (HDFS); data backup; data recovery

## 0 引言

在云计算存储<sup>[1]</sup>时代到来前,网络文件存储采用传统方案:以单机节点为单位、使用基本的文件传输机制(主要包括超文本传输协议(HyperText Transfer Protocol, HTTP)、文件传输协议(File Transfer Protocol, FTP)和网络文件系统(Network File System, NFS)与客户端通信。因旧版本的 NFS 只运行在用户数据包协议(User Datagram Protocol, UDP)上<sup>[2]</sup>,早期文件的网际传输以前两者为主。传统的文件存储具有以下不足:1)可用性(Availability)较低。文件整体存储在一台节点上,为保证可用性,需要采用多机冗余备份的方式。2)扩展性不足。扩展包括单机扩展与冗余节点扩展,扩展成本较大。冗余节点需要人工控制,成本较高,灵活性不足。3)易用性低,故障恢复与存取规模的扩展均需要人工操作。

Hadoop 分布式文件系统(Hadoop Distributed File System, HDFS)<sup>[3]</sup>是 Hadoop 的子项目,是典型的云计算存储方案。HDFS 已有广泛应用,在同类解决方案中具有代表性<sup>[4]</sup>。HDFS 具有云计算存储的优点:1)高可用性。文件被分块,散布于不同的服务器节点上,并自动进行备份。在节点甚至机架失效的情况下,均可恢复文件数据。2)高扩展性。可以实

现动态的节点增删。3)高易用性。数据备份与恢复、服务器节点进入退出集群均被 HDFS 考虑在内<sup>[5]</sup>。同时,HDFS 考虑了文件分布的物理位置,在机架失效的情况下保证文件可恢复。HDFS 也有一定的不足,HDFS 集群包含一个 Namenode 和多个 Datanode, Namenode 失效<sup>[6]</sup>将导致整个集群失效,未来 HDFS 的版本会支持多个 Namenode,但仍对 Namenode 提出较高的要求。

Dynamo<sup>[7]</sup>是 Amazon 的云存储解决方案,主要为 Amazon 的键值(Key-Value, KV)数据库与亚马逊简易存储服务(Amazon Simple Storage Service, Amazon S3)服务器<sup>[8]</sup>提供服务,同样具有云计算存储的优势。Dynamo 采用一致性哈希<sup>[9]</sup>控制文件分块的散布,文件分块的存储复制均在节点之间进行,集群对中心服务器的依存度低。同时,Dynamo 可以设定文件分块的备份数目,以控制可用性。但是 Dynamo 采用简单的备份策略,无法控制文件分块的物理位置,文件分块的多个备份可能存放在同一机架甚至同一节点。在机架失效的情况下,文件可能无法恢复。作为对策,Dynamo 一般备份数目较大,以期“尽可能分布于不同机架”,对磁盘空间的利用效率较低。

本文在深入分析 HDFS 与 Dynamo 的优缺点后,提出一个

收稿日期:2012-07-09;修回日期:2012-08-07。 基金项目:广东省科技厅高新技术产业化科技攻关项目(2011B010200007);成都市科技局创新发展战略研究项目(软科学)(11RKYB016ZF)。

作者简介:陈冬晓(1986-),男,四川德阳人,硕士研究生,主要研究方向:并行计算、云计算;王鹏(1975-),男,四川乐山人,教授,博士生导师,CCF 高级会员,主要研究方向:云计算、并行计算。

改良方案。参考 Dynamo 的一致性哈希以及去中心化思想,来降低集群对中心服务器的依存度;参考 HDFS 的备份策略,改进文件分块的存储备份策略,提高服务器磁盘空间的利用效率。为了使 HDFS 的备份策略可以应用于一致性哈希存储方案,本文方案提出了冲突检测与跳转的思想;为了使磁盘空间利用效率进一步提高,本文提出了文件分块校验备份的方案。通过以上技术的应用,方案在保证可靠性的前提下,充分提高了磁盘空间的利用效率,在保证机架失效文件可恢复的前提下,使磁盘空间占用最小。

## 1 校验编码备份

在数据中心中,节点失效被视为常态,而机架失效可能性较低(机架失效一般由电源设备故障或者网络设备故障造成,此类设别在机房中一般为双备份,主设备发生故障自动启用备用设备)。

HDFS 的典型的三备份策略是:一个文件分块共有三份存储,分别位于当前节点、同一机架不同节点以及不同机架上,分别用于防止节点失效与机架失效(一般为网络设备故障)。

在保证可靠性的前提下,可以进一步减少磁盘空间的占用。作为改进,本文方案使用了校验编码备份:文件块分组,在保存原始文件块的同时,参考海明码编码的方案,以组为单位对文件块校验编码并备份校验编码后的数据。在原始文件块失效时,通过校验编码恢复原始数据。

### 1.1 相关名称约定

在 HDFS 和 Dynamo 将文件分块的基础上,本文方案对文件分块同时进行了分组。

块(Block) 文件在存储时被切割为固定大小的文件块,散布于不同服务器,称为数据块。校验编码生成若干同样大小的文件块,称为校验块。同样整个解决方案中,文件分块大小固定不变。

组(Group) 数据块校验编码时需要分组,多个相继的文件块( $b_1, b_2, b_3, \dots, b_k$ ) 划分为一个数据组,长度用  $k$  表示。用于某一数据组的校验块( $b_{k+1}, b_{k+2}, b_{k+3}, \dots, b_{k+r}$ ) 的组成相应的校验组,长度用  $r$  表示。总的组的长度为  $k+r$ 。

为评价校验编码方案,引入如下指标。

存储占用率 用  $\mu$  表示,代表了校验编码的磁盘开销。其数值越小越好。计算公式为  $\mu = r/k$ 。

通信占用率 用  $\eta$  表示,代表了生成纠错码时的网络通信开销。设参与校验块  $b_{k+i}$  生成的数据块的数目为  $N_{k+i}$ , 则  $\eta = (\sum N_{k+i})/k$ 。

### 1.2 备份过程

校验备份的过程分为如下3步。

1) 客户上传文件数据块。这个步骤中,完成文件的分块、分组以及数据块目标服务器的散布。

2) 数据块目标服务器  $s_i (i = 1, 2, \dots, k)$ , 将数据块传输至预存放相关校验块的服务器  $s_{k+j} (j = 1, 2, \dots, r)$ 。

3) 预存放校验块服务器  $s_{k+j}$  暂存用于校验的数据块,当接收完毕所有所需数据块后,生成相关的校验块。

设校验块  $b_{k+j}$  的第  $x$  位数据为  $b_{k+j}:x$ , 相关的数据组集合为  $\{b_{j_1}, b_{j_2}, \dots, b_{j_k}\}$ , 则生成校验块的公式为:

$$b_{k+j}:x = b_{j_1}:x \oplus b_{j_2}:x \oplus \dots \oplus b_{j_k}:x \quad (1)$$

### 1.3 数据块和校验块的对应关系

数据块和校验块的对应关系类似于海明码中的数据位和

校验位的关系。在给定了  $k$  值后,方案自动确定  $r$  值大小并生成关系矩阵  $group\_map$ :

$$group\_map[i][j] = \begin{cases} T, & b_j \text{ 参与 } p_{k+i} \text{ 的生成} \\ F, & b_j \text{ 不参与 } p_{k+i} \text{ 的生成} \end{cases}$$

$group\_map$  的生成代码如下:

```
group_d_size
group_p_size
group_total_size
i = 1
while k + 1 > 2 ** i - i:
    i += 1
group_d_size = k
group_p_size = i
group_total_size = group_d_size + group_p_size
global group_map
group_map = [[False for t in range(0, group_d_size)] for t in range(0, group_p_size)]
p_sub_list = [2 ** i for i in range(0, group_p_size)]
d_sub_list = []
for sub in range(1, group_total_size + 1):
    if sub in p_sub_list:
        pass
    else:
        col = len(d_sub_list)
        d_sub_list.append(sub)
        temp = sub
        for row in range(0, group_p_size):
            if temp % 2 == 1:
                group_map[row][col] = True
            if temp == 1:
                break
            else:
                temp = temp // 2
```

### 1.4 评价分析

在文件分组过程中,选取的  $k$  值不同,生成的  $r$  值不同,继而影响整个校验备份的效益,如表1所示。

表1 不同  $k$  值下的校验备份评价

$k$	$r$	$\mu$	$\eta$	$\mu\eta$	$k$	$r$	$\mu$	$\eta$	$\mu\eta$
4	3	0.75	2.25	1.69	10	4	0.40	2.40	0.96
5	4	0.80	2.20	1.76	11	4	0.36	2.54	0.93
6	4	0.67	2.17	1.44	12	5	0.42	2.50	1.04
7	4	0.57	2.29	1.31	13	5	0.38	2.46	0.95
8	4	0.50	2.25	1.13	14	5	0.36	2.50	0.89
9	4	0.44	2.33	1.04	15	5	0.33	2.47	0.82

当  $k = 2^i - i - 1 (i \in \{2, 3, 4, 5, \dots\})$  时,  $r = i$ ,  $\mu$  有局部最优解  $\mu = \frac{r}{k} = \frac{i}{2^i - i - 1}$ 。即在不考虑其他情况下,  $i$  值越

大,响应  $k$  值越大,节省的磁盘空间最多。在文件块散布于集群中时,为避免冲突,保证数据可恢复,同组各分块  $d_1, d_2, \dots, d_k, p_{k+1}, p_{k+2}, \dots, p_{k+r}$  需要散布于不同集群。即同时需要满足  $i \leq \lfloor \lg(\text{机架数} + 1) \rfloor$ 。

在实际应用中,应酌情限制  $i$  的大小,以限制数据组长度  $k$ ,原因有两点:

1)  $k$  过大,在2.3节的冲突跳转中,会使跳转过于频繁,增加集群的网络数据流量与服务器负担;

2) 校验块目标服务器缓存数据块与  $k$  正相关,同时增加校验块生成公式的规模,增加服务器校验编码开销。

## 2 冲突判定与跳转

### 2.1 冲突

一致性哈希选用某种哈希算法,分别对文件块以及服务器进行哈希运算,得到相同值空间的哈希值,并将服务器与文件块进行对应作为存取文件块的依据。

为评价文件与服务器的分布策略,作如下定义。

**定义1** 冲突。在机架失效(机架失效隐含了节点失效的情况)的情况下,如果文件不可恢复,则说明文件分布策略与文件可靠性发生了冲突,简称冲突。

一般情况下,使用一致性哈希存储文件块,仅考虑文件块的备份数目。一致性哈希无法保证同一文件块的备份存放在不同机架上,如果同一文件的所有备份处于同一机架,即发生冲突。对于传统方案,采用加大备份数量尽可能规避这种情况。

### 2.2 校验编码备份方案的冲突判定

在使用校验编码备份方案时,为保证节点失效和机架失效情况下文件均可恢复,在文件块与服务器的映射中引入了一定条件。文件拆分为多个分组,不同分组的策略完全相同,所以下面均以单个分组为对象进行分析。

用  $d_i \in s_j$  表示块  $b_i$  保存在服务器  $s_j$  上,用  $s_i \in r_j$  表示服务器  $s_i$  位于机架  $r_j$  上,对于某分组,块存储的服务器集合为  $S = \{s_j | b_i \in s_j (i = 1, 2, \dots, k+r)\}$ , 机架集合  $R = \{r_j | s_i \in r_j (i = 1, 2, 3, \dots, |S|)\}$ 。

数据机房的集群状况较为复杂,网络时延<sup>[10]</sup>和策略的简明性都需要考虑在内,使用了两段式冲突判定,先检验本地冲突,本地无冲突再检验跨服务器的冲突。

1) 本地的冲突判定。如果同组的两个分块处于同一服务器  $\exists b_i, b_j: b_i \in s_i \& b_j \in s_i (i, j = 1, 2, \dots, k+r)$ , 即对于单组文件块,  $|s| > 1$  时,则发生冲突。服务器  $s_i$  失效将导致文件不可恢复。

2) 跨服务器的冲突判定。 $\exists d_i \in s_i, p_i \in s_j, s_i \in r_i \& s_j \in r_i (i, j = 1, 2, \dots, k+r)$ , 即对于单组文件块存放服务器,  $|r| > 1$  时,则发生冲突。机架  $r_i$  失效将导致文件不可恢复。

### 2.3 异步冲突跳转

#### 2.3.1 冲突跳转

冲突发生后,冲突所在服务器将文件块传输至一致性哈希的下一节点,以期消除冲突。直接传输文件块,而不事先查询目标服务器是否能够解决冲突,这是从网络利用效率和实现复杂性上考虑的。集群内部网络为高速网络<sup>[11]</sup>,典型速度在 100 Mbps 或者 1000 Mbps。直接传输文件块的成本更低。在选用文件分块为 1 MB 大小时,传输时延与请求查询目标服务器并返回结果的总耗时相当<sup>[12]</sup>。下一节点接收到文件块后,重新检测冲突。这一过程持续到冲突消除。

跳转过程中产生的冲突文件块保存在服务器中,可用于分散集群负载,待服务器磁盘空间不足时再进行清理。

以图1为例,A文件块存储于1-1节点;B文件存储于1-2节点,发生冲突,B文件块跳转至1-2的下一节点4-1,冲突消失;C文件块存储于1-1节点,发生冲突,跳转至2-2节点,冲突消失。

#### 2.3.2 异步处理方式

在冲突发生后,服务器采用了异步的处理方式:服务器暂

时忽略冲突,完成后续的文件块传输、存储、编码校验的工作,以上工作完成后,服务器再将文件块转移至其他服务器,新服务器重新检测冲突,直到冲突消失。

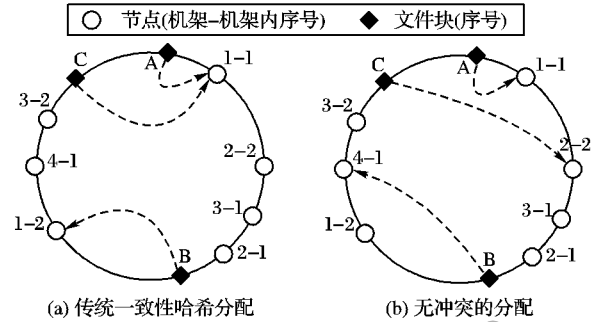


图1 冲突跳转示意

采用异步冲突跳转,具有以下优点:

1) 使内部的冲突跳转对客户透明化,保持客户端上传文件的流程简洁性,进而简化了客户端的复杂度与通信过程。在整个存储过程中,集群与客户端的通信时延是最大的,简化集群和客户端的通信过程可以显著提高存储作业速度。

2) 增加了集群的协调性。校验编码备份涉及集群中多个节点的协同工作,单一节点搁置冲突,继续传输存储编码校验流程,以维持与其他节点的同步性,同时降低了节点策略的复杂性,减小开发难度,增加程序稳定性。

## 3 模拟验证

本文方案已经实现了模拟运行对校验编码备份和冲突跳转策略进行验证。模拟配置如下:数据组长度  $k = 8 (r = 4)$ , 集群机架数目 20, 每机架节点数目 2, 服务器总数 40。

方案实现了状态图表展示程序,可以实时表示文件分块在集群中的状态。

图2展示了冲突跳转之前的文件块分布,服务器实现了文件的分块、文件分块在集群的散布以及校验编码备份。

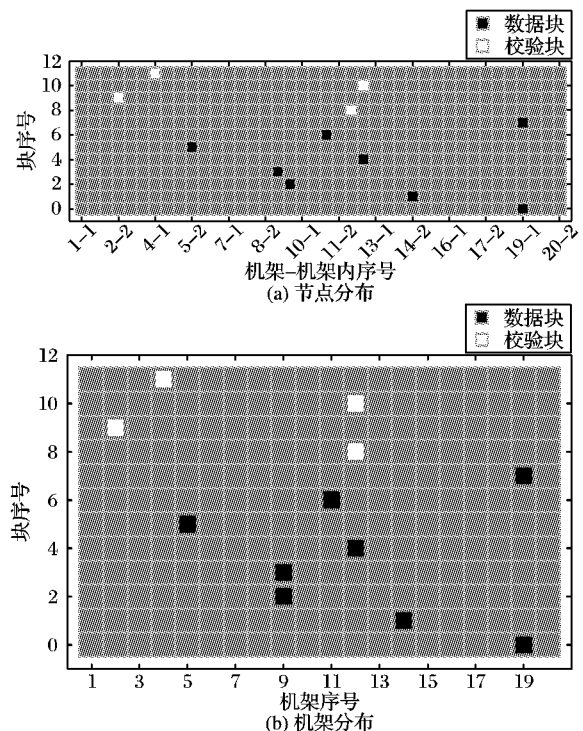


图2 冲突跳转前文件块分布

测试文件大小为 8 MB, 校验数据占用磁盘空间 4 MB, 存

储占用率 $\mu=0.5$ 。HDFS下文件分块存储,以64 MB大小的文件为例,备份数据大小为128 MB,折算的存储占用率 $\mu=2$ 。Dynamo的存储占用率 $\mu$ 与系统配置参数有关,若备份系数为3,则存储占用率同为2,与HDFS相同。

在生成校验数据时,将产生网络通信开销。校验块生成需要数据块的参与。参照式(1)与`group_map`生成代码,校验块数据的生成共需要传输18个数据块,而生成一份镜像备份只需要传输8个数据块,通信占用率 $\eta=18/8=2.25$ 。HDFS与Dynamo均生成镜像备份,在典型情况下,生成两份备份,通信占用率 $\eta=2$ 。

与一次性的通信开销相比较,磁盘开销是长期的。本文方案以瞬时的通信开销为代价,获取了持久性的磁盘空间上的收益。

图2仅展示了校验编码的生成。校验编码生成后,第9机架第1节点、第19机架第1节点、第12机架第2节点和第12机架第1节点发生了冲突。在第9机架和第12机架失效时,文件将不可恢复。

图3展示了冲突跳转后的文件块分布。冲突跳转完成后,数据块 $d_1, d_2, \dots, d_8$ 以及校验块 $p_1, p_2, p_3, p_4$ 散布在不同的机架上。在任意机架失效的情况下,丢失的文件块可通过式(1)进行恢复。由表2可得,任何机架失效,其数据均可由其他机架的数据恢复,存储的可靠性得到了保证。

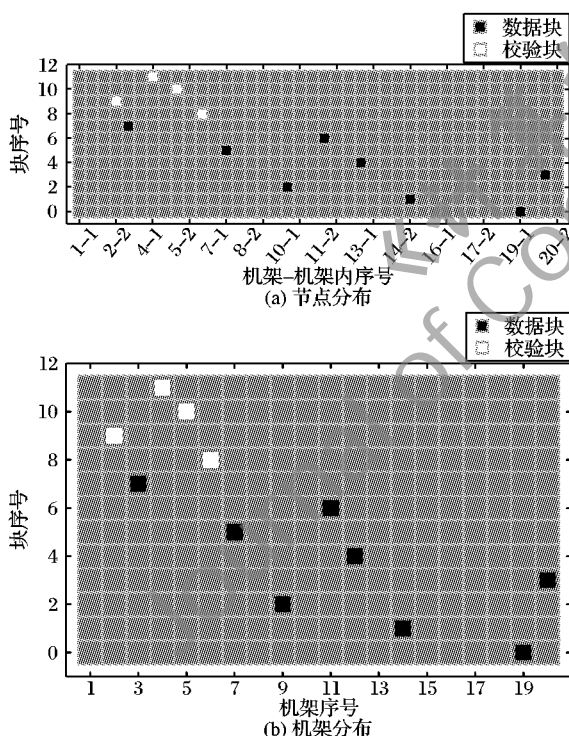


图3 冲突跳转后文件块分布

## 4 结语

模拟运行实现了文件分块、服务器散列、校验备份、冲突检验和跳转等关键功能,验证了本文方案的可行性。

在保证可靠性的前提下,校验编码减少了对磁盘空间和核心服务器性能的需求。

校验编码节省存储空间与 $k$ 值的选取有关,模拟运行中 $k=8$ ,与HDFS相比,节省了约62.5%的磁盘空间。如果选择较大的 $k$ ,将进一步减少空间占用。

模拟运行中,12个文件块累积产生了9次冲突,对集群负载有一定影响,如果在实践中实现异步的冲突跳转方式,负载将被均摊到较长的时间段中,对集群的影响较小。冲突跳转的次数与 $k$ 值、集群规模、文件块的散布均匀程度有关。

本文方案使用MD5校验码生成一致性哈希值,哈希值分布并不均匀,较为均匀的哈希值可以均衡系统负载,并且可以减少冲突跳转的次数。提高哈希值分布的均匀程度,可以通过改进哈希值算法或其他方式<sup>[13]</sup>实现。

表2 测试文件可靠性分析

失效机架	失效文件块	恢复用文件块	恢复用机架
1	—	—	—
2	校验块9	0, 2, 3, 4, 6	19, 9, 20, 12, 11
3	数据块7	4, 5, 6, 11	12, 7, 11, 4
4	校验块11	4, 5, 6, 7	12, 7, 11, 3
5	校验块10	1, 2, 3, 7	14, 9, 20, 3
6	校验块8	0, 1, 3, 4, 6	19, 14, 20, 12, 11
7	数据块5	4, 6, 7, 11	12, 11, 3, 4
8	—	—	—
9	数据块2	1, 3, 7, 10	14, 20, 3, 5
10	—	—	—
11	数据块6	4, 5, 7, 11	12, 7, 3, 4
12	数据块4	5, 6, 7, 11	7, 11, 3, 4
13	—	—	—
14	数据块1	2, 3, 7, 10	9, 20, 3, 5
15	—	—	—
16	—	—	—
17	—	—	—
18	—	—	—
19	数据块0	1, 3, 4, 6, 8	14, 20, 12, 11, 6
20	数据块3	1, 2, 7, 10	14, 9, 3, 5

## 参考文献:

- [1] Storage Networking Industry Association (SNIA). Cloud storage reference model: Version 0.3 rev [EB/OL]. [2012-03-15]. <http://www.snia.org/sites/default/files/CloudStorageReferenceModelV03.pdf>.
- [2] CALLAGHAN B. NFS over RDMA [EB/OL]. [2012-04-05]. <http://static.usenix.org/events/fast02/wips/callagcal.pdf>.
- [3] BORTHAKUR D. HDFS architecture guide [EB/OL]. [2012-03-18]. [http://hadoop.apache.org/common/docs/r0.20.0/hdfs\\_design.pdf](http://hadoop.apache.org/common/docs/r0.20.0/hdfs_design.pdf).
- [4] SHVACHKO K, KUANG H, RADIA S, et al. The Hadoop distributed file system [C]// Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies. Washington, DC: IEEE Computer Society, 2010: 1-10.
- [5] 许春玲, 张广泉. 分布式文件系统 Hadoop HDFS 与传统文件系统 Linux FS 的比较与分析[J]. 苏州大学学报: 工科版, 2010, 30(4): 5-19.
- [6] WHITE T. HDFS reliability [EB/OL]. [2012-04-20]. [http://www.cloudera.com/wp-content/uploads/2010/03/HDFS\\_Reliability.pdf](http://www.cloudera.com/wp-content/uploads/2010/03/HDFS_Reliability.pdf).
- [7] DECANDIA G, HASTORUN D, JAMPANI M, et al. Dynamo: Amazon's highly available key-value store [C]// SOSP'07: Proceedings of the 21st ACM Symposium on Operating Systems Principles. New York: ACM Press, 2007: 205-220.

### 3.2 速度验证

本研究用4台戴尔PC机(2.80 GHz,双核)组成的集群对并行算法的加速比<sup>[11-12]</sup>和效率进行验证。其中1台PC机既作为主节点,也作为从节点,内存为2 GB,另外3台PC机内存为1 GB。分布式平台采用C++语言开发的Sector/

Sphere-2.6,统一配置Ubuntu 10.04 32位操作系统。为了测试并行算法的性能,分别采用100,200,300,400个水稻代谢样本作为数据集,数据预处理及解卷积用Leco公司的Chroma TOF software软件完成,解卷积之后单个样本所含信号峰数为200~500。具体实验数据如表1所示。

表1 并行算法实验数据

样本数	运行时间/s				加速比				效率			
	1个节点	2个节点	3个节点	4个节点	1个节点	2个节点	3个节点	4个节点	1个节点	2个节点	3个节点	4个节点
100	2760	1632.6	1200.0	960.0	1	1.69	2.30	2.88	1	0.85	0.77	0.72
200	9180	5355.6	3841.2	3080.4	1	1.71	2.39	2.98	1	0.86	0.80	0.75
300	18780	10731.6	7307.4	5887.2	1	1.75	2.57	3.19	1	0.87	0.86	0.80
400	31620	17469.6	11932.2	9610.8	1	1.81	2.65	3.29	1	0.91	0.88	0.82

从表1中可以看出:当样本数为100时,用4个节点的集群测试并行算法,加速比只有2.88,效率仅为0.72;当节点数不变,增大数据量,并行算法的加速比随之提升,效率随之提高;当样本数达到400时,加速比达到了3.29,效率达到了0.82。这是因为数据量较小时,数据通信时间以及算法串行部分执行时间所占比重较大,集群的利用率低;随着数据量的增大,各节点的计算性能得到更充分的利用,并行算法的优势逐渐体现。当数据量一定时,算法的加速比随着节点数的增加而增大,但并行效率逐渐降低,这主要是因为随着节点数的增加,总执行时间减少,串行部分执行时间不变,串行部分的执行时间占总执行时间比重增加。

### 4 结语

Sector/Sphere平台是用C++语言搭建的开源高效数据云平台,将GC-MS数据处理流程移植于Sector/Sphere平台实现并行化,能够解决因GC-MS数据量大、处理过程复杂导致过长处理时间的问题。本研究设计了基于Sector/Sphere平台的GC-MS数据处理并行框架,并实现了基于此平台的多样本并行对齐算法,未来的研究方向可考虑将GC-MS数据处理的其他过程(如去噪、基线校正、解卷积等)的算法移植于Sector/Sphere平台,并实现并行化。

#### 参考文献:

- [1] TRYGG J, HOLMES E, LUNDSTEDT T. Chemometrics in metabolomics [J]. *Journal of Proteome Research*, 2007, 6(2): 469 - 479.
- [2] KOH Y, PASIKANTI K K, YAP C W, *et al.* Comparative evaluation of software for retention time alignment of gas chromatography/time-of-flight mass spectrometry-based metabolomic data [J]. *Journal of Chromatography A*, 2010, 1217(52): 8308 - 8316.
- [3] LOMMEN A. MetAlign: interface-driven, versatile metabolomics

tool for hyphenated full-scan mass spectrometry data preprocessing [J]. *Analytical Chemistry*, 2009, 81(8): 3079 - 3086.

- [4] KATAJAMAA M, MIETTINEN J, ORESIC M. MZmine: toolbox for processing and visualization of mass spectrometry based molecular profile data [J]. *Bioinformatics*, 2006, 22(5): 634 - 636.
- [5] LUEDEMANN A, STRASSBURG K, ERBAN A, *et al.* TagFinder for the quantitative analysis of gas chromatography-mass spectrometry (GC-MS)-based metabolite profiling experiments [J]. *Bioinformatics*, 2008, 24(5): 732 - 737.
- [6] LANGMEAD B, SCHATZ M C, LIN J, *et al.* Searching for SNPs with cloud computing [J]. *Genome Biology*, 2009, 10(11): 134.
- [7] GU Y H, GROSSMAN R. Sector and sphere: the design and implementation of a high performance data cloud [J]. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 2009, 367(1897): 2429 - 2445.
- [8] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters [J]. *Communications of the ACM*, 2005, 51(1): 107 - 113.
- [9] ROBINSON M D, de SOUZA D P, KEEN W W, *et al.* A dynamic programming approach for the alignment of signal peaks in multiple gas chromatography-mass spectrometry experiments [J]. *BMC Bioinformatics*, 2007, 8: 419.
- [10] de SOUZA D P, SAUNDERS E C, McCONVILLE M J. Progressive peak clustering in GC-MS Metabolomic experiments applied to Leishmania parasites [J]. *Bioinformatics*, 2006, 22(11): 1391 - 1396.
- [11] ZHAO W Z, MA H F, HE Q. Parallel K-means clustering based on MapReduce [C]// *CloudCom'09: Proceedings of the 1st International Conference on Cloud Computing*. Berlin: Springer-Verlag, 2009: 674 - 679.
- [12] XU X W, JAGER J, KRIEGER H P. A fast parallel clustering algorithm for large spatial databases [J]. *Data Mining and Knowledge Discovery*, 1999, 3(3): 263 - 290.

(上接第214页)

- [8] Amazon.com Inc. Amazon simple storage service (Amazon S3) [EB/OL]. [2012-03-15]. <http://aws.amazon.com/s3>.
- [9] LEWIN D. Consistent hashing and random trees: algorithms for caching in distributed networks [D]. Cambridge, Massachusetts: Massachusetts Institute of Technology, 1998.
- [10] PAI V S, ARON M, BANGA G, *et al.* Locality-aware request distribution in cluster-based network servers [C]// *Proceedings of the 8th International Conference on Architectural Support for Programming Lan-*

- guages and Operating Systems*. New York: ACM Press, 1998: 205 - 216.
- [11] 中国移动通信集团设计院有限公司. 综合布线系统工程设计规范, GB50311—2007 [S/OL]. [2012-04-22]. <http://www.sc-cic.org.cn/zywyh/GB50311.htm>.
- [12] KUROSE J F, ROSS K W. Computer networking: a top-down approach [M]. 4th ed. 北京: 高等教育出版社, 2010.
- [13] 周敬利, 周正达. 改进的云存储系统数据分布策略[J]. *计算机应用*, 2012, 32(2): 309 - 312.