基于可用性度量的分布式文件系统节点失效恢复算法

廖 彬[1,2] 于 炯[1] 钱育蓉[1] 杨兴耀[2]

(新疆大学软件学院 乌鲁木齐 830008)[1] (新疆大学信息科学与工程学院 乌鲁木齐 830046)[2]

摘 要 现有分布式文件系统中处理节点失效时采用的恢复策略耗费较多的带宽与磁盘空间资源,且影响系统的稳定性。通过研究分布式文件系统HDFS集群结构、数据块存储机制、节点与数据块状态之间的关系,定义了集群节点矩阵、节点状态矩阵、文件分块矩阵、数据块存储矩阵与数据块状态矩阵为度量数据块可用性建立了基础数据模型。在实现数据块可用性度量基础上,设计了基于可用性度量的节点失效恢复算法并分析了算法的性能。实验结果表明:新算法在保证系统中所有数据块可用性的前提下比原恢复策略减少了恢复所需带宽与磁盘资源,缩短了节点恢复时间,提高了系统稳定性。

关键词 云计算,分布式文件系统,失效恢复,可用性度量

中图法分类号 TP311 **文献标识码** A

Node Failure Recovery Algorithm for Distributed File System Based on Measurement of Data Availability

LIAO Bin[1,2] YU Jiong[1] QIAN Yu-rong[1] YANG Xing-yao[2]

(School of Software, Xinjiang University, Urumqi 830008, China)[1]

(School of Information Science and Engineering, Xinjiang University, Urumqi 830046, China)[2]

Abstract The strategy for distributed file system dealing with node failure needs much bandwidth and disk space resources and affects stability of the system. By studying HDFS's cluster structure, data blocks storage mechanism, the state relationship between node and block, we defined the cluster nodes matrix, node status matrix, file block partition matrix, block storage matrix and block state matrix. Those definitions enable us to model the availability of data block easily. Based on the measurement of data block's availability, we proposed the new node failure recovery algorithm and analyzed the performance of the algorithm. The experimental results show that compared with the original strategy, the new algorithm ensures the availability of all blocks in the system and reduces the bandwidth and disk space resources for recovery, shorts the recovery time, and improvs the stability of system.

Keywords Cloud computing, Distributed file system, Failure recovery, Measurement of data availability

1 引言

Web2.0、Web Services、SaaS与虚拟化技术等的不断发展使得云计算得到越来越多的应用,作为云计算底层数据基础管理平台,分布式文件系发挥着重要的作用。目前比较著名的分布式文件系统有Google公司的GFS(Google File System)[1,4]开源的HDFS(Hadoop Distributed File System)[2]、Lustre、MooseFs以及清华大学自主研发的CarrierFs等。其中GFS管理着Google公司百万服务器上的海量数据,基于GFS的分布式数据库BigTable[3]支撑着Goolge搜索、地图、社交网络等服务。HDFS为Hadoop[2]底层分布式文件系统,由于Hadoop能够部署在通用平台上,并且具有可扩展性(Scalable)、低成本(Economical)、高效性(Efficient)与可靠性(Reliable)等优点,使其在分布式计算领域得到了广泛的运用,并且已逐渐成为工业与学术界事实上的海量数据并行处理标准。考虑到HDFS的开放性与其广泛的运用,本文选取了HDFS为研究对象,但本文算法同样适用于GFS、Lustre、MooseFs、CarrierFs等类似系统。本文研究分布式文件系统的节点失效恢复算法,节点失效算法的主要作用是保证系统中数据的可靠性与可用性,以避免节点失效引起的数据不可用对上层应用的影响。而现有分布式文件系统中处理节点失效策略需耗费较多的带宽与磁盘空间,且影响系统的稳定性。所以,本文在研究HDFS集群结构、数据块存储机制、节点与数据块状态之间的关系基础上,定义了DataNode节点矩阵、节点状态矩阵、文件分块矩阵、数据块存储矩阵、数据块状态矩阵与HDFS集群可用性度量矩阵,其中HDFS集群可用性度量矩阵用于度量HDFS集群中所有文件与数据块的可用性。在实现数据块可用性度量的基础上设计了新的节点失效

到稿日期:2012-03-21 返修日期:2012-07-29 本文受国家自然科学基金项目(60863003,61063042),新疆维吾尔自治区自然科学基金项目(2011211A011)资助。

廖 彬(1986—),男,博士生,主要研究方向为数据库技术、网格与云计算,E-mail:liaobin@xju.edu.cn;于 炯(1964—),男,博士,教授,博士生导师,主要研究方向为网络安全、网格与分布式计算;钱育蓉(1980—),女,博士后,副教授,主要研究方向为计算机应用;杨兴耀(1984—),男,博士生,主要研究方向为分布式计算、网格计算。

恢复算法，克服了原恢复策略需要占用较多带宽与磁盘空间资源的缺点，并且缩短了节点恢复时间，提高了系统稳定性。

本文第 1 节介绍相关背景；第 2 节分析研究现状与现有的节点失效恢复策略；第 3 节提出 HDFS 下数据块可用性度量方法；第 4 节提出基于可用性度量的分布式文件系统节点失效算法的具体步骤；第 5 节建立算法的性能分析模型；第 6 节通过实验验证了算法的有效性；最后对全文进行了总结。

2 研究现状与节点失效恢复策略

现有研究大多集中在提高存储系统可靠性上，文献[5-8]研究了基于 RAID 的存储系统可靠性保证方法；文献[9-14]研究了基于 P2P 的存储系统可靠性保证策略。现有存储系统失效恢复方面的研究比较少，其中文献[15-17]分别采用 distributed sparing、decluster、DATUM 等方法减少节点失效恢复时所需要的资源与时间。本文则是研究分布式文件系统中的节点失效，在保证系统数据可用性前提下改进现有恢复策略，以达到缩短恢复时间、减少恢复资源消耗的目的。而文献[19-22]中虽然对分布式文件系统进行节能计算研究时提及了数据块可用性概念，但并没有明确给出理论定义与证明，且没有将其用于节点失效处理之中。

HDFS 分布式文件系统是 Hadoop 框架的一部分，负责数据的分布式存储及管理。HDFS 系统由一个 NameNode 与多个 DataNode 组成，其中 NameNode 维护系统元数据信息（如命名空间，数据块存储及映射信息，系统配置信息等），而 DataNode 中以数据块的形式存储着系统中的真正数据，并通过副本策略提高数据的可靠性。Data-Node定期的以发送心跳(heartbeat)的形式将自身信息与状态报告发送给 NameNode，当 NameNode 不能收到某个节点心跳信息时便认为该 DataNode 处于失效状态，此时 NameNode 将对该节点进行失效恢复处理以保证系统中数据的可靠性。HDFS 在进行节点失效恢复时进行数据块复制操作，当任意节点失效时，NameNode 会将存储于该节点上所有低于副本系数的数据块重新复制一份放入其它活动 DataNode 中。

节点失效算法主要作用是保证系统中数据的可靠性与可用性，本文在保证数据可靠性与可用性的前提下，在对数据的可用性建模的基础上设计了新的节点失效恢复算法。根据第 3 节定理 2 可知，当系统中不可用节点数量小于系统配置的副本系数 m 时，系统中所有数据块仍然处于可用状态，此时并不需要立即进行节点恢复操作，因为有些节点在失效不久可能因为故障的排除而重新恢复正常工作，此时 NameNode 还需要进行数据块一致性检查与多余数据块删除操作（节点恢复时由复制操作产生的），再一次耗费系统资源。根据定理 2 知，只有当系统中不可用节点数量大于等于系统配置的副本系数 m 时才需要立即进行恢复操作，并且不需要复制失效节点上所有的数据块，而只需要恢复少量处于不可用状态的数据块。由此可见，新的节点失效恢复算法相比传统的恢复算法将节省大量的系统资源，同时提高系统的稳定性。

3 数据块可用性度量

3.1 相关数据模型定义

本节通过对分布式文件系统 HDFS 集群结构、数据块存储机制、节点与数据块状态之间的关系的研究，定义了集群节点矩阵、节点状态矩阵、文件分块矩阵、数据块存储矩阵、数据块状态矩阵，并在此基础上证明了文件数据块的可用性条件，为 HDFS 下的文件与数据块的可用性提供了度量方法。

HDFS 集群一般由多个机架 RACK 组成，而一个 RACK 内部又由多个服务器组成，并且 HDFS 集群通常由一个 NameNode 和多个 DataNode 节点构成，因此将 HDFS 集群节点结构表示为矩阵，如定义 1 所示。

定义 1（集群节点矩阵） 设 HDFS 集群由 r 个 RACK 组成，并且设所有 RACK 中都有 s 个 DataNode 节点服务器，用 dn 表示 DataNode 节点服务器，将 HDFS 集群中的所有 DataNode 节点表示为矩阵 C_{s*r}：

$$C_{s*r}=\begin{bmatrix} dn_{11} & dn_{12} & \cdots & dn_{1r} \\ dn_{21} & dn_{22} & \cdots & dn_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ dn_{s1} & dn_{s2} & \cdots & dn_{sr} \end{bmatrix} \tag{1}$$

式中，$dn_{ij}\in C_{s*r}(1\leqslant i\leqslant s,1\leqslant j\leqslant r)$ 表示编号为 j 的 RACK 中第 i 个 DataNode 节点服务器。设 $k=s*r$ 表示集群中 DataNode 节点服务器的数量。

定义 2（节点状态矩阵） HDFS 集群中的 DataNode 节点可能存在多种状态，设状态标识集合为 $state=\{0,1\}$，其中 0 表示该 DataNode 节点服务器处于不可用状态（即失效状态），1 表示 DataNode 节点处于可用状态。在定义 1 中的 C_{s*r} 结构基础上，根据每个 DataNode 节点的当前状态构建节点状态矩阵 DS_{s*r} 为：

$$DS_{s*r}=\begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1r} \\ s_{21} & s_{22} & \cdots & s_{2r} \\ \vdots & \vdots & \ddots & \vdots \\ s_{s1} & s_{s2} & \cdots & s_{sr} \end{bmatrix} \tag{2}$$

式中，$s_{mn}\in\{0,1\}(1\leqslant m\leqslant s,1\leqslant n\leqslant r)$。

定义 3（文件分块矩阵） HDFS 集群中文件都被拆分成一系列的数据块存放在 DataNode 节点中，并采用在同一个集群中存储多个副本的策略来提高数据的可靠性。在由 $k(k>3)$ 个 DataNode$\{dn\in C_{s*r}\}$ 组成的 HDFS 集群中，设某数据块副本系数为 m，数据块大小为 bs 的文件 F 大小为 $n*bs$，那么根据 HDFS 存储与副本机制 F 会有 $n*m$ 个数据块随机存储在 k 个 DataNode 节点中，其文件分块可由 F_{n*m} 矩阵表示。其中，原文件 F 由数据块 $\{b_{11},b_{21},\cdots,b_{n1}\}$ 组成，而矩阵 $F_{n*(m-1)}$ 表示该文件的副本，n 表示组成文件 F 的原始数据块数量。

$$F_{n*m}=\begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nm} \end{bmatrix} \tag{3}$$

$$F_{n*(m-1)}=\begin{bmatrix} b_{12} & \cdots & b_{1m} \\ b_{22} & \cdots & b_{2m} \\ \vdots & \ddots & \vdots \\ b_{n2} & \cdots & b_{nm} \end{bmatrix} \tag{4}$$

另外，HDFS 机架感知的数据块存储策略如图 1 所示。当用户向 HDFS 集群上传数据时，第一个数据块 b_{11} 随机存放在某个 DataNode 节点中，第二个数据块 b_{12}（b_{11} 的副本 1）存放在与 b_{11} 不同的机架上的 DataNode 节点中，第三个数据块 b_{13}（b_{11} 的副本 2）存放在与 b_{12} 相同的机架但不同的 DataNode

节点中。如果副本系数 $m>3$，其余的数据块就随机地存放在除 b_{11}、b_{12}、b_{13} 存储节点以外的任意 DataNode 节点中。由此可见，基于机架感知的数据块存储策略使得数据块的存储位置具有较大的随机性。

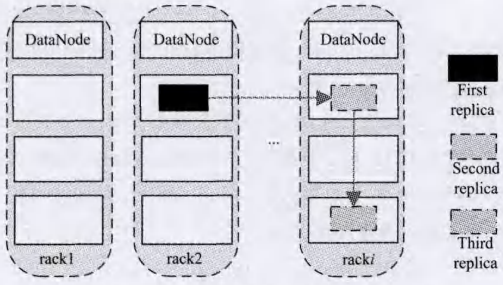

图 1　基于机架感知的数据块存储策略

定义 4(数据块存储矩阵)　文件 F 首先根据矩阵 F_{n*m} 对文件进行分块，然后根据机架感知的数据块存储策略将数据块存储于 HDFS 集群中的 DataNode 节点上，即文件分块矩阵 F_{n*m} 中的 $n*m$ 个数据块 b_{ij} ($1\leqslant i\leqslant n, 1\leqslant j\leqslant m$)存储于矩阵 $dn\in C_{s*r}$ 中的 k 个 DataNode 服务器中，并遵循一个数据块 b_{ij} 只能存储在一个 DataNode 节点中，b_{ij} 与其副本数据块不能存储在同一个 DataNode 节点中的规律。那么，将 F_{n*m} 中的数据块符号 b_{ij} ($1\leqslant i\leqslant n, 1\leqslant j\leqslant m$)替换为存放该数据块的 DataNode 节点标识 $dn\in C_{s*r}$，将得到的新矩阵定义为数据块存储矩阵 FS_{n*m}：

$$FS_{n*m}=\begin{bmatrix} bd_{11} & bd_{12} & \cdots & bd_{1m} \\ bd_{21} & bd_{22} & \cdots & bd_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ bd_{n1} & bd_{n2} & \cdots & bd_{nm} \end{bmatrix} \tag{5}$$

式中，$bd_{ij}\in C_{s*r}$ ($1\leqslant i\leqslant n, 1\leqslant j\leqslant m$)表示 F_{n*m} 中数据块 b_{ij} 存储的 DataNode 节点标识，即数据块 b_{ij} 存储于 DataNode 节点 dn_{ij} 中。例如，当 F_{n*m} 中数据块 b_{13} 存放在 dn_{45} 中时，bd_{13} 值为 dn_{45}。机架感知的存储策略使得 FS_{n*m} 中的任意一行均不能存在相同的元素。通过数据块存储矩阵 FS_{n*m} 可以快速地定位任意文件任意数据块的存放位置。

定义 5(数据块状态矩阵)　将定义 4 中数据块存储矩阵 FS_{n*m} 中的节点表示符号 bd 替换为节点状态矩阵 DS_{s*i} 中相应的状态，并用符号 bs 表示，将新的矩阵定义为数据块状态矩阵 BS_{n*m}：

$$BS_{n*m}=\begin{bmatrix} bs_{11} & bs_{12} & \cdots & bs_{1m} \\ bs_{21} & bs_{22} & \cdots & bs_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ bs_{n1} & bs_{n2} & \cdots & bs_{nm} \end{bmatrix} \tag{6}$$

式中，$bs_{ij}\in\{0,1\}$ ($1\leqslant i\leqslant n, 1\leqslant j\leqslant m$)。状态 1 表示该数据块存储在活动的 DataNode 节点中，数据块处于可用状态；0 表示该数据块存储在不可用(失效)的 DataNode 节点中。数据块状态矩阵 BS_{n*m} 可用于判定文件 F 的可用性。

定理 1　HDFS 集群中的任意文件 F 处于可用状态的条件是该文件数据块状态矩阵 BS_{n*m} 中每一行中至少存在一个状态为 1 的数据块。

证明：用反证法证明。

假设文件 F 的数据块状态矩阵 BS_{n*m} 中存在第 i ($1\leqslant i\leqslant n$)行 $\{bs_{i1}, bs_{i2}, \cdots, bs_{im}\}$ 中不存在状态为 1 的数据块，即 $\{bs_{i1}, bs_{i2}, \cdots, bs_{im}\}\in\{0\}$ ($1\leqslant i\leqslant n$)。

根据定义 3 与定义 5 可知，此时文件 F 的 n 个分块中第 i 个数据块与其副本都处于不可用的 DataNode 节点中，数据块 $\{bs_{i1}, bs_{i2}, \cdots, bs_{im}\}$ 都处于不可用状态。

用户读取文件 F 时，因为数据块 $\{bs_{i1}, bs_{i2}, \cdots, bs_{im}\}$ 都处于不可用状态，n 个数据块中的第 i 个数据块将读取失败，造成不能将文件 F 组合成功返回给用户。此时，文件 F 对于读取用户不可用。证毕。

定理 2　在不改变任意数据块原始存储结构的前提下，设集群中不可用节点数为 u，所有文件副本系数为 m，当 $u<m$ 时，集群中所有文件处于可用状态；当 $u\geqslant m$ 时，集群中所有文件处于可用状态的概率为：

$$(1-\frac{A(u,m_1)}{A(k,m_1)})^{n_1}*(1-\frac{A(u,m_2)}{A(k,m_2)})^{n_2}*(1-\frac{A(u,m_3)}{A(k,m_3)})^{n_3}*\cdots*(1-\frac{A(u,m_w)}{A(k,m_w)})^{n_w}$$

式中，$n_1, n_2, n_3, \cdots, n_w$ 分别表示集群中的 w 个文件 $files=\{F_1, F_2, \cdots, F_w\}$ 的原始分块数，$m_1, m_2, m_3, \cdots, m_w$ 分别表示 w 个文件的副本系数。

证明：(1)当 $u\leqslant m$ 时，由于任意数据块 b_{ij} 与其副本数据块不能存储在同一个 DataNode 节点中，因此任意数据块的 m 个备份不可能同时处于不可用状态，即任意文件的数据块状态矩阵 BS_{n*m} 中任意行都满足定理 1，此时集群中所有文件都处于可用状态。

(2)在拥有 k 个 DataNode 节点的 HDFS 集群中有 u 个不可用的节点，根据定理 1，当同一数据块的 m 个备份同时存储于不可用节点中时，该数据块处于不可用状态，此事件发生的概率为：

$$\frac{A(u,m)}{A(k,m)} \tag{7}$$

反之，该数据块可用的概率为：

$$1-\frac{A(u,m)}{A(k,m)} \tag{8}$$

文件 F 由 n 个数据块组成，当 n 个数据块都可用时，文件 F 可用，此事件发生的概率为：

$$(1-\frac{A(u,m)}{A(k,m)})^{n} \tag{9}$$

集群中所有文件处于可用状态即表示集群中任意文件 $\{F_1, F_2, \cdots, F_w\}$ 都满足定理 1，亦即集合 $files=\{F_1, F_2, \cdots, F_w\}$ 中所有文件都可用，此事件发生的概率为：

$$(1-\frac{A(u,m_1)}{A(k,m_1)})^{n_1}*(1-\frac{A(u,m_2)}{A(k,m_2)})^{n_2}*(1-\frac{A(u,m_3)}{A(k,m_3)})^{n_3}*\cdots*(1-\frac{A(u,m_w)}{A(k,m_w)})^{n_w} \tag{10}$$

3.2　建模示例

图 2 所示为由 4 个 RACK 组成并且每个 RACK 中由 4 个节点服务器组成的 HDFS 集群。根据定义 1 可将集群节点表示为：

$$C_{4*4}=\begin{bmatrix} dn_{11} & dn_{12} & dn_{13} & dn_{14} \\ dn_{21} & dn_{22} & dn_{23} & dn_{24} \\ dn_{31} & dn_{32} & dn_{33} & dn_{34} \\ dn_{41} & dn_{42} & dn_{43} & dn_{44} \end{bmatrix}$$

设文件 $F1$ 分块矩阵为 $F1_{3*3}$，在图 2 集群中数据块存储

矩阵为 $FS_{F1_{3*3}}$；文件 F2 分块矩阵为 $F2_{2*3}$，在图 2 集群中数据块存储矩阵为 $FS_{F2_{2*3}}$；文件 F3 分块矩阵为 $F3_{3*2}$，在图 2 集群中数据块存储矩阵为 $FS_{F3_{3*2}}$。

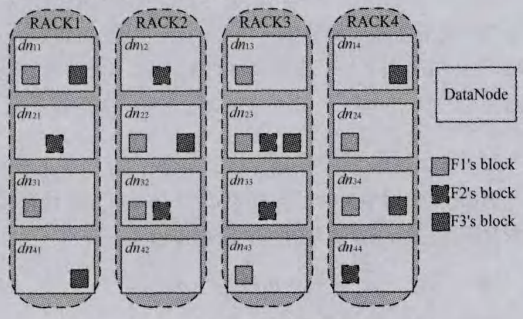

图 2　数据块存储模型示例

$$F1_{3*3}=\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \quad FS_{F1_{3*3}}=\begin{bmatrix} dn_{11} & dn_{13} & dn_{24} \\ dn_{22} & dn_{23} & dn_{34} \\ dn_{31} & dn_{32} & dn_{43} \end{bmatrix}$$

$$F2_{2*3}=\begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \quad FS_{F2_{2*3}}=\begin{bmatrix} dn_{21} & dn_{12} \\ dn_{32} & dn_{23} \\ dn_{33} & dn_{44} \end{bmatrix}$$

$$F3_{3*2}=\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \end{bmatrix} \quad FS_{F3_{3*2}}=\begin{bmatrix} dn_{11} & dn_{23} & dn_{34} \\ dn_{41} & dn_{22} & dn_{14} \end{bmatrix}$$

由图 2 与数据块存储矩阵可以看出，集群中任一节点失效都不会影响到数据块的可用性。当节点 dn_{11}、dn_{22}、dn_{23}、dn_{34} 同时失效时，3 个文件数据块状态矩阵变化为：

$$BS_{F1_{3*3}}=\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \rightarrow BS_{F1_{3*3}}=\begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$BS_{F2_{2*3}}=\begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \rightarrow BS_{F2_{2*3}}=\begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$$

$$BS_{F3_{3*2}}=\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \rightarrow BS_{F3_{3*2}}=\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

由此可得，当节点 dn_{11}、dn_{22}、dn_{23}、dn_{34} 同时失效时，造成了文件 F1 与 F2 数据块的不可用。

4　基于可用性度量的节点失效恢复算法

如图 3 所示，本文提出的基于可用性度量的节点失效恢复算法可分为以下几个步骤：1）构建基础数据；2）当 DataNode 失效时首先进行数据块可用性度量，验证节点的失效是否引起数据块的不可用；3）当上一步验证节点的失效引起数据块的不可用时，通过复制备份方法恢复相应的数据块，否则结束。

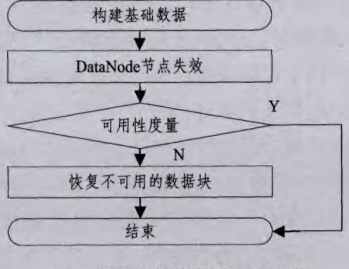

图 3　节点失效恢复算法流程图

4.1　构建基础数据

基于可用性度量的分布式文件系统节点失效算法需要构建的初始化参数与基础数据如下：

（1）集群信息，其中包括节点矩阵 C_{s*i} 与节点状态矩阵 DS_{s*i}。

（2）集群中所有文件的集合 $files=\{F_1,F_2,\cdots,F_w\}$，其中 $w=|files|$ 表示文件的个数，$(n_1,n_2,n_3,\cdots,n_w)$ 分别表示 w 个文件的原始分块数，$(m_1,m_2,m_3,\cdots,m_w)$ 分别表示 w 个文件的副本系数，所有文件的分块大小为 bs，单位为兆（M）。

（3）文件相关信息矩阵的集合，其中包括：集群中所有文件数据块存储矩阵的集合 $fileSMs=\{S_1,S_2,\cdots,S_w\}$；所有文件数据块状态矩阵的集合 $fileBSMs=\{BS_1,BS_2,\cdots,BS_w\}$。

（4）恢复数据块集合 recoveryBlocks 中存放需要进行失效恢复的数据块标识。

为了更好地存储与组织上述节点失效算法需要的基础数据，本文为 HDFS 加入了 RecoveryNode 节点来专门负责集群的失效恢复工作（见图 4）。RecoveryNode 节点中主要存储集群节点与状态矩阵、数据块存储矩阵、数据块状态矩阵，恢复数据块集合 recoveryBlocks 等相关信息。RecoveryNode 节点执行节点失效的主要步骤为：

（1）DataNode 向 NameNode 发送心跳信息报告当前所处状态。

（2）当 NameNode 不能接收到任意 DataNode 节点心跳时，NameNode 认定该 DataNode 处于失效状态并将该节点相关信息（主要是该节点上所有数据块的信息）发送给 RecoveryNode 节点。

（3）RecoveryNode 节点按照第 4.2 节与第 4.3 节中的相关步骤确认需要进行恢复的数据块。

（4）将 recoveryBlocks 集合发送给 NameNode 节点执行恢复操作。

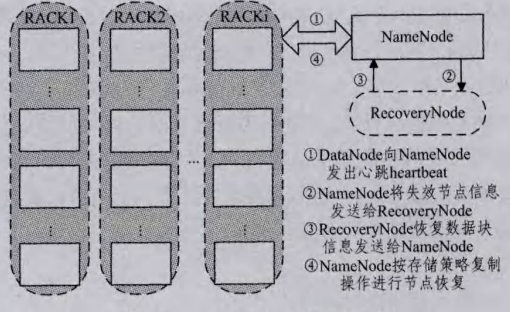

图 4　RecoveryNode 与集群的关系

4.2　数据块可用性度量

当任意 DataNode 节点服务器 dn 在节点状态矩阵 DS_{s*i} 中由状态 1 转化为 0，即由活动状态转化为失效状态时，数据块可用性度量方法由以下算法完成：

（1）根据 NameNode 发送给 RecoveryNode 节点关于失效节点的信息得到失效节点 dn 中所有数据块的信息。

（2）循环更新失效节点 dn 中所有数据块的数据块状态矩阵 BS_{n*m} 中对应的数据块状态为 0。

（3）根据定理 1 检查步骤（2）BS_{n*m} 中数据块的可用性，如果满足可用性要求，则跳转步骤（2）循环继续检查下一数据块所属数据块状态矩阵 BS_{n*m} 的可用性。如果不满足可用性要求，将此数据块标识存放在集合 recoveryBlocks 中。

(4)循环结束。

值得注意的是,在步骤(3)中可以对可用性要求进行设置,如集群中数据块至少保存有 1 个或者 2 个任意数据块的副本。

4.3 节点失效恢复

本节通过复制备份方法恢复因失效引起的处于不可用状态的数据块。具体由以下算法完成:

(1)当 recoveryBlocks 集合为空时,结束节点失效恢复算法。

(2)当 recoveryBlocks 集合不为空时,循环取出其中处于不可用状态的数据块标识。查询该数据块对应的数据块存储矩阵 S_{n*m},并通过复制的方法进行失效恢复。

(3)当循环恢复完成 recoveryBlocks 集合中所有数据块时,结束节点失效恢复算法。

5 算法性能分析模型

本节给出失效恢复算法的性能分析模型,其中符号定义说明如表 1 所列。

表 1 符号定义说明表

符号	定义
k	HDFS 集群中 DataNode 节点的数量
bs	数据块分块大小
m	数据块副本系数
bn	失效 DataNode 节点中存储数据块的数量
d	HDFS 集群内 RACK 间的数据传输速率
t	失效 DataNode 节点恢复时间
sp	失效 DataNode 节点恢复所需磁盘容量
u	HDFS 集群中同时失效节点的数量
T	同时失效节点为 d 时恢复时间
SP	同时失效节点为 d 时恢复所需磁盘容量

5.1 原恢复策略

首先分析原恢复策略,在拥有 k 个 DataNode 节点的 HDFS 集群中,设节点上存储的数据块大小为 bs 且存储有 bn 个数据块,集群内 RACK 间的数据传输速率为 u。由于 HDFS 基于机架感知的存储策略使得在同一 RACK 内不存在相同的数据块,因此在进行失效恢复时数据块的复制传输全部在 RACK 之间而不是同在一 RACK 内部进行。当集群中失效节点数为 1 时,失效需要恢复该节点上所有的数据块,所以失效恢复需要的磁盘容量为:

$$sp=bn*bs \tag{11}$$

失效恢复所需时间为:

$$t=(bn*bs)/d \tag{12}$$

当集群中失效节点数为 u 时,失效恢复所需的磁盘容量为:

$$SP=\sum_{i}^{u}bn_i*b_s(1\leqslant i\leqslant u) \tag{13}$$

式中,bn_i 表示 u 个失效节点中第 i 个节点上存储数据块的数量。

失效恢复所需时间为:

$$T=(\sum_{i}^{u}bn_i*b_s)/d(1\leqslant i\leqslant u) \tag{14}$$

5.2 基于可用性度量的失效恢复策略

由本文第 3 节定理 2 可知,当 $u<m$ 时,集群中所有文件均处于可用状态,即当集群中失效节点数小于数据块副本系数时,节点的失效并不会影响集群中数据块的可用性;当 $u\geqslant m$ 时,u 个失效节点上数据块的总和为:

$$\sum_{i}^{u}bn_i(1\leqslant i\leqslant u) \tag{15}$$

由定理 2 的证明过程可知,u 个节点同时失效时引起任意数据块失效的概率如第 3 节式(3)所示。那么由式(3)与式(11)可计算出,当 u 个节点同时失效时引起不可用数据块的数量为:

$$\sum_{i}^{u}bn_i*\frac{A(u,m)}{A(k,m)}(1\leqslant i\leqslant u) \tag{16}$$

那么,当集群中失效节点数为 u 时,基于可用性度量的失效恢复策略进行失效恢复时所需磁盘容量为:

$$SP=\sum_{i}^{u}bn_i*\frac{A(u,m)}{A(k,m)}*bs(1\leqslant i\leqslant u) \tag{17}$$

所需失效恢复所需时间为:

$$T=(\sum_{i}^{u}bn_i*\frac{A(u,m)}{A(k,m)}*bs)/d(1\leqslant i\leqslant u) \tag{18}$$

当 u 个失效节点同时存在于同一 RACK 中时,由于 HDFS 基于机架感知的存储策略使得在同一 RACK 内不存在相同的数据块,因此这种情况并不会影响任意数据块的可用性,即在新的恢复策略中并不需要实施恢复操作。

6 实验评价与比较

基于第 5 节所述性能分析模型,本文使用 MATLAB、cloudsim[18] 实现对算法的分析,并通过性能评价与比较来验证基于可用性度量失效恢复算法的有效性。

图 5 所示为不可用 DataNode 节点数与 HDFS 集群数据块可用性之间的关系,其中横轴表示不可用节点的数量,纵轴表示集群中所有数据块可用的概率。其 line1 表示集群中节点数 $k=100$、副本系数 $m=3$、集群中原数据块个数为 10000 时,不可用状态的节点数量与整个系统可用性概率之间的关系。line1 与 line2 比较,当 k 与 n 条件相同时,副本系数 m 值越大,HDFS 集群可用概率越高;line2 与 line3 比较,当 m 与 n 条件相等时,DataNode 节点数越大,HDFS 集群可用概率越高;line1 与 line4 比较,当 m 与 k 条件相等时,n 越小,HDFS 集群可用概率越高。

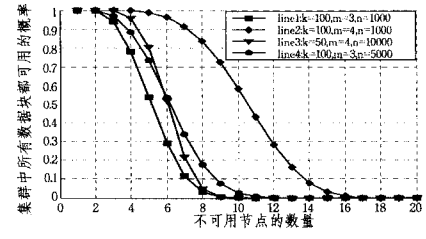

图 5 不可用节点数与所有数据块可用性之间的关系

设集群中节点数量 $k=10$,100G 数据按照数据块大小 $bs=64$M,$m=2$ 或 3 时采用机架感知的存储策略随机存储到集群中,可以计算出每个节点平均存储数据块 $bn=160$。如图 6 所示,在相同条件下,当失效节点数量较小时,基于可用性度量的节点失效恢复策略在进行节点恢复时需恢复的数据块数量远远小于原全恢复策略;但随着失效节点数的增大,两种恢复策略都需要恢复大量的数据块并且恢复效率逐渐缩小。相同条件下,当副本系数 m 由 2 提高到 3 时,所需恢复的数据块数量减少。结合第 5 节公式可以看出,随着所需恢复数据块数量的减少,相比原全恢复策略,基于可用性的恢复策略将大大节省磁盘空间与失效恢复所需的时间。

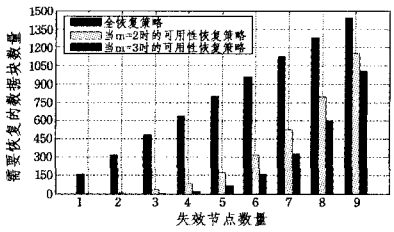

图 6　不同恢复策略下的恢复成本比较

图 7 所示为 $bn=200$ 时，集群节点 k 与副本系数 m 对节点恢复数据块数量的影响。从图 7 可以看出：相同条件下，副本系数 m 值越大，所需恢复数据块数量越小；集群节点数量越大，所需恢复数据块数量越小；需恢复数据块的数量随着失效节点数的增加而增加。

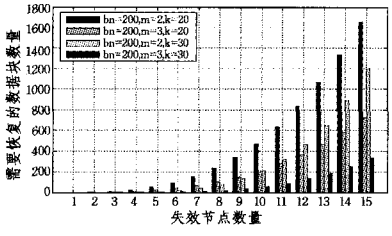

图 7　不同恢复策略下的恢复成本比较

结束语　本文在分析 HDFS 集群结构、数据块存储机制、节点与数据块状态之间的关系的基础上，定义了集群节点矩阵、节点状态矩阵、文件分块矩阵、数据块存储矩阵与数据块状态矩阵，为度量数据块可用性提供了基础数据模型。考虑到现有节点失效恢复策略耗费带宽与磁盘资源并且影响系统稳定性的缺点，提出了基于可用性度量的分布式文件系统节点失效恢复算法，当节点失效时并不立即进行失效处理，而是首先考察失效节点上数据块的可用性是否受到影响。算法以保证数据块的可用性为原则，新算法与原策略相比，大大减小了恢复时复制的数据块数量，减少了的恢复时所需带宽与磁盘资源，缩短了恢复时间。值得注意的是，基于可用性的恢复策略还受到集群节点数 k 与副本系数 m 两个参数的影响。下一步工作主要是将本文对分布式文件系统数据块可用性度量运用到节能计算领域，设计出基于数据块可用性度量的节能支持算法，基于数据块可用性度量的节能存储结构改进等方面。

参 考 文 献

[1] Ghemawat S, Gobioff H, Leung S T. The Google File System[C]// Proceedings of 19th ACM Symposium on Operating System Principles (SOSP2003). New York, USA, 2003: 29-43

[2] Borthaku D. The Hadoop Distributed File System: Architecture and Design [OL]. http://hadoop. apache. org/common/docs/r0. 18. 2/ hdfs_design. pdf, 2007-07-01

[3] Chang F, Dean J, Ghemawat S, e t al. Bigtable: A Distributed Storage System for Structured Data[C]// Proceedings of the 7th Symposium on Operating Systems Design and Implementation (OSDI). Seattle, WA, USA, 2006: 205-218

[4] Dean J, Ghemawat S. MapReduce: Simplifed data processing on large clusters[C]// Proceedings of the Conference on Operating System Design and Implementation (OSDI). San Francisco, USA, 2004: 137-150

[5] Chen P M, Lee E K, Gibson G, et al. RAID: High-performance, reliable secondary storage[J]. ACM Computing Surveys, 1994, 26(2): 145-185

[6] Burkhard W A, Menon J. Disk array storage system reliability [C]// Proceedings of the 23rd International Symposium on Fault-Tolerant Computing(FTCS '93). 1993: 432-441

[7] Plank J S. A tutorial on Reed-Solomon coding for fault tolerance in RAID-like systems [J]. Software Practice and Experience (SPE), 1997, 27(9): 995-1012

[8] Schwarz T J. Generalized Reed Solomon codes for erasure correction in SDDS[C]// Workshop on Distributed Data and Structures(WDAS 2002). Paris, Mar. 2002

[9] Rhea S, Eaton P, Geels D, et al. Pond: the OceanStore prototype [C]// Proceedings of the 2003 Conference on File and Storage Technologies(FAST). 2003: 1-14

[10] Weatherspoon H, Kubiatowicz J. Erasure coding vs. replication: A quantitative comparison[C]// Proceedings of the First International Workshop on Peer-to-Peer Systems (IPTPS 2002). Cambridge, Massachusetts, 2002

[11] Rowstron A, Druschel P. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility[C]// Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01). Banff, Canada, 2001: 188-201

[12] Dabek F, Kaashoek M F, Karger D, et al. Wide-area cooperative storage with CFS[C]// Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01). Banff, Canada, Oct. 2001: 202-215

[13] Ripeanu M, Iamnitchi A, Foster I. Mapping the Gnutella network[J]. IEEE Internet Computing, 2002, 6(1): 50-57

[14] Muthitacharoen A, Morris R, Gil T M, et al. Ivy: A read/write peer-to-peer file system[C]// Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI). Boston, MA, Dec. 2002

[15] Alvarez G A, Burkhard W A, Cristian F. Tolerating multiple failures in RAID architectures with optimal storage and uniform declustering[C]// Proceedings of the 24th International Symposium on Computer Architecture. Denver, CO, 1997: 62-72

[16] Menon J, Mattson R L. Distributed sparing in disk arrays[C]// Proceedings of Compcon92. 1992, 92: 410-416

[17] Muntz R R, Lui J C S. Performance analysis of disk arrays under failure[C]// Proceedings of the 16th Conference on Very Large Databases (VLDB). 1990: 162-173

[18] Calheiros R N, Ranjan R, Beloglazov A, et al. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms [J]. Software: Practice and Experience, 2010, 41(1): 23-50

[19] Leverich J, Kozyrakis C. On the energy (in)efficiency of hadoop clusters[J]. ACM SIGOPS Operating Systems Review, 2010, 44 (1): 61-65

[20] Maheshwari N, Nanduri R, Varma V. Dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework[J]. Future Generation Computer Systems, 2011, 28(1): 119-127

[21] Kaushik R T, Bhandarkar M. GreenHDFS: towards an energy-conserving, storage-efficient, hybrid hadoop compute cluster[C]// Proceedings of the 2010 international conference on Power aware computing and systems. Berkeley, USA, 2010: 1-9

[22] Kaushik R T, Bhandarkar M, Nahrstedt K. Evaluation and analysis of GreenHDFS: A self-adaptive, energy-conserving variant of the hadoop distributed file system[C]// Proceedings of the 2nd IEEE International Conference on Cloud Computing Technology and Science. Indianapolis, USA, 2010: 274-287