**密级： 保密期限：**

xm 拷贝

**硕士学位论文**



**题目： 集群管理系统关键技术研究**

**学 号： 2015110003**

**姓 名： 许巡枝**

**专 业： 信息与通信工程**

**导 师： 别红霞**

**学 院： 信息与通信工程学院**

**2017年01月15日**



**A Thesis for Master Degree**

**TITLE: RESEARCH OF CLUSTER**

**MANAGEMENT SYSTEM**

|  |  |
| --- | --- |
| **Student No.:** | **2015110003** |
| **Author:** | **Xunzhi Xu** |
| **Major:** | **Information and Communi-**  **cation Engineering** |
| **Supervisor:** | **Hongxia Bie** |
| **School:** | **School of Information and Communication Engineering** |

**Jan. 15th, 2018**

独创性（或创新性）声明

本人声明所呈交的论文是本人在导师指导下进行的研究工作及取得的研究成果。尽我所知，除了文中特别加以标注和致谢中所罗列的内容以外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得北京邮电大学或其他教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示了谢意。

申请学位论文与资料若有不实之处，本人承担一切相关责任。

本人签名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 日期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

关于论文使用授权的说明

学位论文作者完全了解北京邮电大学有关保留和使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属北京邮电大学。学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许学位论文被查阅和借阅；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。（保密的学位论文在解密后遵守此规定）

保密论文注释：本学位论文属于保密在年解密后适用本授权书。

非保密论文注释：本学位论文不属于保密范围，适用本授权书。

本人签名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 日期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

导师签名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 日期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**集群管理系统关键技术研究**

**摘 要**

随着互联网技术的快速发展，各种数据量呈现出爆炸性的增长，通过单机对海量数据进行分析处理已经难以进行，分布式集群系统成海量数据存储与分析处理的必然趋势。分布式系统将多台机器组合成一个系统，对外提供服务，在满足海量数据存储、计算和分析的同时，能够有效降低对单台服务器的性能要求及成本。随着数据规模的急剧增长，集群的规模不断扩大，集群的功能日愈复杂，集群管理的难度增大，集群系统的可靠性和一致性成为影响集群性能的关键。如何设计可靠高效的集群管理机制，一直是业内重要的研究课题之一。

论文基于海量结构化数据的列式存储机制，深入研究集群管理系统的关键算法，以提高集群管理系统的效率及可靠性，设计并实现集群管理系统的关键功能。

论文为降低系统网络通信复杂度、减少集群系统本身负载，深入研究了传统的Master节点选举算法和心跳检测模型，提出了一套低通信开销的Master节点选举机制，降低了选举过程中的通信复杂度，设计了一种心跳检测机制，将心跳信息和集群节点信息同步相结合，从而简化集群系统结构，提高集群网络数据传输效率。论文的主要工作如下：

（1）深入研究集群系统Master节点选举算法，提出一种基于节点性能指标的低通信开销的选举机制。集群系统需要一个主控节点来维护集群节点信息并分配外部任务。选举算法不仅要在集群建立时选举出性能最高的节点来担任主控节点，还需要在原来的主控节点故障时，快速地选出性能最优的节点来接管主控节点的工作。

（2）设计了一种高效率的双向心跳检测机制。心跳机制是保证集群检测节点故障检测并及时处理的最基本模块。为了提高网络传输效率，论文在深入研究传统心跳检测模型的基础上，将集群信息同步和心跳检测相结合，实现双向心跳检测，既实现了信息的同步，又能提高网络传输效率。

（3）基于本文研究的集群Master选举算法和心跳检测机制，设计实现集群管理系统的关键功能模块，主要包括Master 节点选举模块、心跳检测和信息表同步模块、节点上线检测模块以及故障处理模块等。

**关键词：**集群管理 选举算法 心跳检测

注：摘要要能够准确反映论文内容。而且在表达上要与论文后面的章节保持一致。

之前的一些用语经不起推敲， 而且没有很严谨清楚地描述出论文的工作。

论文工作的每一段总结， 都要能够说清楚你的这个工作 为什么做， 怎么做的 结果如何， 而不是过程描述。 例如你的心跳检测机制是的特点， 要让读者通过摘要能够看到，而不是你那种过程性的描述， 你的描述只说了你有一种心跳检测模块， 并看不出你的模块到底是什么思想， 怎么做的。

此外，关于关键内容， 整篇论文的前后说法一定要一致， 例如 心跳检测机制，是思想部分， 那可以都叫机制好了。后面实现的部分， 可以是模块。

诸如此类， 请注意

请按照以上逻辑， 对全文的表达方式及逻辑再整理一下。

**RESEARCH OF CLUSTER**

**MANAGEMENT SYSTEM**

**ABSTRACT**

With the rapid development of the Internet technology, the amount of data produced in the network also shows an explosive growth, to store and analyze these data are facing many challenges. It has been difficult to analyze massive data by a single machine, Distributed systems have become an effective solution to mass data processing. Distributed systems reduce the system's hardware costs by combining multiple inexpensive PC units into a single system that provides services to meet the needs of storing, computing and analyzing of mass data. However, with the rapid growth of data size, the scale of the cluster is also expanding and the functions of the cluster are increasingly complicated, which increases the difficulty of cluster management and poses a great challenge to the reliability and consistency of the cluster system. Therefore, how to provide a reliable and efficient management mechanism for clusters has always been one of the important topics.

Based on the column-based storage of massive structured data, this paper deeply studies the existing cluster management system and its key technologies, on the basis of explaining the basic principle of its key technologies, the key function modules of cluster management system are researched and realized. Starting from reducing the complexity of system network communication and reducing the load of the system itself, the paper deeply studies the traditional election algorithm and the traditional heartbeat detection model, puts forward a set of election mechanism with low communication overhead, reduces the communication complexity generated in the election process, The heartbeat module is redesigned and combine the heartbeat information with the synchronization of cluster node information to simplify the system structure and improve the network transmission efficiency. In this paper, the main works is as follow:

(1) In-depth study of cluster election algorithm, put forward a low communication overhead election mechanism based on node performance. The cluster needs a master node to maintain cluster node information and assign external tasks. The election algorithm not only needs to elect the node with the highest performance to serve as the master node, but also needs to quickly select the optimal node to take over the work of the master node when the original master node fails.

(2) Redesigned cluster heartbeat module. The heartbeat module is the most basic module to ensure cluster detection node failure and timely processing. In order to improve the efficiency of network transmission, based on the further study of the traditional heartbeat detection model, this paper combines the synchronization of cluster information and heartbeat detection to redesign the heartbeat module.

(3) Taking the election algorithm and heartbeat detection mechanism as the core, the key functional modules of the cluster management system are designed and implemented.

**KEY WORDS:** Cluster management Election algorithm Heartbeat detection

英文摘要，写得实在太不像英文了！！要意译， 专业英语一定是很严谨的，

建议找一篇博士学位论文的英文摘要， 或者国外论文的英文， 基本语句仿写一下

**目 录**

[第一章 绪论 1](#_Toc498953122)

[1.1 课题背景与研究意义 1](#_Toc498953123)

[1.2 集群管理技术发展现状 2](#_Toc498953124)

[1.3 课题主要研究内容 3](#_Toc498953125)

[1.4 论文组织结构 4](#_Toc498953126)

[1.5 本章小结 4](#_Toc498953127)

[第二章 集群管理关键技术研究 5](#_Toc498953128)

[2.1 集群技术 5](#_Toc498953129)

[2.1.1 集群类型 5](#_Toc498953130)

[2.1.2 集群的特征 6](#_Toc498953131)

[2.2 经典的分布式选举算法 9](#_Toc498953132)

[2.2.1 欺负选举算法 9](#_Toc498953133)

[2.2.2 欺负选举算法的改进算法 11](#_Toc498953134)

[2.3 心跳检测技术 14](#_Toc498953135)

[2.3.1 心跳检测模型 14](#_Toc498953136)

[2.3.2 HeartBeat项目 16](#_Toc498953137)

[2.4 本章小结 18](#_Toc498953138)

[第三章 基于节点性能的低通信开销选举算法 19](#_Toc498953139)

[3.1 节点性能评估 19](#_Toc498953140)

[3.2 让贤与继承机制 20](#_Toc498953141)

[3.2.1 让贤机制 20](#_Toc498953142)

[3.2.2 继承机制 22](#_Toc498953143)

[3.3 性能分析与仿真 23](#_Toc498953144)

[3.3.1 让贤机制通信复杂度 23](#_Toc498953145)

[3.3.2 继承机制通信复杂度 24](#_Toc498953146)

[3.3.3 仿真结果 25](#_Toc498953147)

[3.4 本章小结 28](#_Toc498953148)

[第四章 结合同步信息的心跳检测机制 29](#_Toc498953149)

[4.1 双向心跳检测机制 29](#_Toc498953150)

[4.2 同步信息 30](#_Toc498953151)

[4.3 结合同步信息的双向心跳检测机制 32](#_Toc498953152)

[4.4 性能分析与仿真 34](#_Toc498953153)

[4.4.1 通信开销 34](#_Toc498953154)

[4.4.2 网络传输效率 37](#_Toc498953155)

[4.5 本章小结 39](#_Toc498953156)

[第五章 集群管理系统分析与设计 40](#_Toc498953157)

[5.1 应用场景 40](#_Toc498953158)

[5.2 集群管理系统拓扑结构 40](#_Toc498953159)

[5.3 系统总体功能规划 43](#_Toc498953160)

[5.3.1 选举模块 45](#_Toc498953161)

[5.3.2 心跳检测与信息同步模块 47](#_Toc498953162)

[5.3.3 节点上线处理模块 49](#_Toc498953163)

[5.3.4 节点故障处理机制 51](#_Toc498953164)

[5.4 功能测试与分析 52](#_Toc498953165)

[5.4.1 测试目的 52](#_Toc498953166)

[5.4.2 测试方法 52](#_Toc498953167)

[5.4.3 测试结果 53](#_Toc498953168)

[5.5 本章小结 55](#_Toc498953169)

[第六章 结论与展望 56](#_Toc498953170)

[6.1 全文总结 56](#_Toc498953171)

[6.2 展望 56](#_Toc498953172)

[参考文献 58](#_Toc498953173)

[致谢 61](#_Toc498953174)

[攻读学位期间取得的研究成果 62](#_Toc498953175)

1. 绪论
   1. 课题背景与研究意义

近些年来，随着计算机科学技术和社会的不断发展，互联网已经深入影响着人们在学习、生活和工作的方方面面，伴随而来的是大量数据信息的产生，尤其是物联网技术和人工智能领域的兴起，需要对大量的图像、视频、社交网络内容以及穿戴设备数据等信息进行处理，使计算机需要进行存储和计算分析处理的数据量呈现出指数型的增长[2]。以视频存储为例，一小时的DV格式的视频素材大约消耗12GB的存储空间，一些未经过压缩的视频占用的空间更大，一分钟标准清晰度画面的视频大小需要2GB的存储空间，一分钟非压缩的1920×1080像素的高清晰度视频的大小跟是达到了9.38GB[3]。而对企业和科研机构而言，这些数据往往具有巨大的潜在价值，如何对海量数据进行有效、可靠的存储管理和分析计算成为一个大的挑战。

在传统模式下，通过纵向扩展来应对计算资源增长的需求，即采用中型机替代小型机，大型机替代中型机的方式来扩大计算机本身的计算资源。但是，由于大型机价格昂贵，而且维护费用高，导致其性价比比较低，难以普及推广，传统的单机模式已经难以应对大数据时代的存储和计算需求。使用普通机器组成的集群系统由于性价比高等原因，已经取代传统的大型机或巨型机，得到广泛应用[4]。集群技术对多台单机服务器进行资源的统一管理和调度，以一个整体的方式对外提供服务，同时，集群技术也支持横向拓展，能够满足用户庞大的计算能力的需求，是当前计算机领域实现超大规模运算的重要发展方向之一。

集群是指通过网络互连，可协作执行某个任务的独立计算机集合，其主要包含两个方面的含义：第一，从硬件角度来看，集群中的每一台计算机都是独立自主的；第二，从软件的角度来看，外部用户可以将整个集群系统当作一台计算机[5]。集群技术通过并行计算技术，能提供更高的数据处理速度，例如对海量数据进行计算分析，通过利用多台服务器进行并发工作，能在较短的时间周期内完成同等计算量的运算工作。近年来，随着云存储、云计算和云服务等概念的兴起和相关技术的迅猛发展，集群技术凭借其可靠性、可拓展性、可用性、高效性以及较高的性价比等特点，再度成为企业和科研机构的热门研究方向之一。

随着集群技术的不断发展及其在各行各业的广泛应用，对集群系统的可靠性和稳定性等性能也提出了更高的要求。许多服务都要求集群系统具备高可靠性，即在部分软硬件发生故障时，系统仍然能继续正常运行，对外提供服务，同时也不会造成数据的丢失和损坏。目前，互联网公司不仅要求其集群系统能7×24小时持续地对外提供服务，还要求集群在容错和故障处理上具备较高的可靠性，因为一旦系统中断服务或者是数据发生丢失，都可能会带来灾难性的后果，造成无法弥补的重大损失[7]。据戴尔公司发布的新闻，其平均每天在网站上的交易额为1400万美元，一个小时的服务中断都会造成大约五十八万美元的损失[7]，如果是发生数据的丢失和损坏，其损失更是巨大。

由此可见，如何提高集群系统的可靠性，保证系统对外提供连续、稳定的服务是当前计算机集群技术的重要研究方向，同时也是极具研究价值的课题之一。

* 1. 集群管理技术发展现状

集群管理技术并不是近几年才出现的，其发展起点可以追溯到1983年由Digital公司研制出的VAX cluster，它通过允许多台计算机访问同一个共享磁盘的方式来实现负载均衡，而且还支持单个系统镜像[7]，在1996年，应用于Windows NT平台的高可用集群服务器管理软件问世，微软公司还为其制定了相应的工业标准，对应用程序接口进行统一管理。与此同时，NCR公司也发布了基于Windows NT的LifeKeeper2.0。LifeKeeper2.0能够给用户提供一个拥有完全容灾机制的解决方案，并且支持具有16个服务器节点的集群系统，在信息数据、服务和资源方面实现了高可用性，此外，LifeKeeper2.0也加入了故障检测和现场恢复机制，当集群主节点发生故障时，LifeKeeper2.0一方面保护资源；另一方面会将相应的服务切换到一台具有绝对优先权的服务器上，同时对资源进行转移，在进行服务切换时，集群系统会出现短时间的休眠，等待切换结束，集群系统继续对外服务[8]。2008年，NEC公司发布了具有高可用性的集群管理软件——ExpressCluster X系列，该软件在远程容灾、故障检测、系统维护操作和虚拟化技术等方面进行改善，而且通过对软硬件资源的实时监控，能准确无误地发现系统故障，借此来保证集群系统能在最短的时间内进行服务切换、恢复正常运行。

在国内，一些企业为了满足市场需求，也致力于研发高性能、高可用的集群系统。联想公司在1999年发布一款高性能的计算机集群，该集群的节点全部采用低成本的小型机器，在当时，具备和该集群相当的计算能力的机器成本大约是该集群成本的2到4倍；同年，我国某集群网络公司也推出了一款Linux安全集群系统，通过了公安部的安全认证，目前该公司的集群技术仍然在多个网站中应用，如Linux网站[www.linux.com](http://www.linux.com)和瑞士电信等，该集群技术的核心代码也被纳入美国RedHat发行版；由中国自主研发的高性能计算机集群“自强2000-SUHPCS”于2000年9月在上海大学诞生；阿里巴巴公司的服务器集群“飞天”的集群规模在2013年8月突破了5000台，成为世界首家对外提供5K云计算服务能力的公司，其集群规模比肩FaceBook和google等世界顶级互联网公司[9]。此外，国内还有一些高可用集群系统诞生，如联鼎软件公司发布的LanderCluster软件、朗新公司推出的LongShine Cluster Sever以及联想公司发布的NS10000高性能集群服务器等。

有上述内容可见，高可用性集群系统在国内外都有着良好的发展氛围。随着大数据和人工智能领域的发展，集群种类越来多，其规模也越来越庞大，各企事业单位和科研机构越来越重视高可用集群技术的应用，整体而言，高可用集群系统的应用领域越来越广泛，其支持的平台逐渐多样化，并且对集群性能、可靠性和稳定性也提出了更高的要求，如何保证集群的可靠性和稳定性，降低系统复杂度，提高集群系统性能，一直以来都是各大企业、科研机构和学者的重点研究方向之一。

* 1. 课题主要研究内容

本课题研究了现有的集群管理系统，分析了其存在的问题，借鉴国内外的先进研究成果，对项目进行了充分的需求分析，从保证系统的可用性和可靠性出发，以降低系统内部通信开销为目的，重点研究了低通信开销的选举算法和结合同步信息的心跳检测技术，降低系统内部的通信量，对相应算法的性能进行理论推导和仿真分析，完成集群管理系统的设计，并对关键模块进行实现。

本课题的主要研究内容有：

（1）基于节点性能的低通信开销选举算法。选举算法的目的是为集群选举出一台性能最好的机器来担任集群的主控节点（Master节点），维护集群各个节点状态，是保障集群状态一致性的基础。论文深入研究了传统的欺负选举算法，针对其算法通信复杂度高、通信量波动较大的问题，提出了一种基于节点性能的低通信开销选举算法，该算法以节点性能表为核心，分为让贤和继承两大机制，能有效地降低系统的通信量。

（2）结合同步信息的双向心跳检测机制。心跳检测机制是一种进行网络连接故障情况检测的技术，在集群中，心跳检测机制通过主控节点和普通节点的心跳模块之间互相发送心跳信息，以此检测网络连接或者节点机器是否正常，是集群及时发现节点故障，并及时进行相应的故障处理的前提保证，同时也是为集群系统可靠性提供保障的一个重要模块。论文在深入研究传统的PULL&PUSH心跳检测机制的基础上，参考HeartBeat开源项目，结合本集群的具体特点和需求，设计了一种结合同步信息的心跳检测机制，能有效检测节点故障，同时提高网络传输效率。

（3）集群管理系统关键模块的设计与实现。以上述的选举算法和心跳检测机制为核心，设计并实现了高可用的集群管理系统，并对系统功能和关键模块功能进行测试。

* 1. 论文组织结构

根据论文主要工作和研究内容，本文结构安排如下：

第一章介绍了课题研究背景和研究意义，简要介绍了集群技术的发展过程及其现状，引出课题主要研究内容，并给出论文结构。

第二章介绍了集群的基本概念及其关键技术，包括集群的类型和特点、传统的欺负选举算法和心跳检测技术。

第三章深入讨论了本课题的关键算法——基于节点性能的低通信开销选举算法，并与欺负选举算法进行仿真对比分析。

第四章讨论了集群的心跳检测技术和信息同步技术，给出一种结合同步信息的双向心跳检测机制，并对其网络传输效率进行分析。

第五章介绍了集群管理系统及其关键模块的实现，首先介绍集群的系统架构，然后详细介绍其关键模块的详细设计流程。

第六章对论文工作进行总结，并指出今后工作中的优化和改进之处。

* 1. 本章小结

本章主要阐述了论文的选题背景和研究意义、国内外对集群管理技术的研究以及应用现状，随后介绍了论文的主要研究内容和论文的整体组织结构。

1. 集群管理关键技术研究
   1. 集群技术

集群，通常也被称作分布式系统，是指由一组自治的计算机组成的集合，这组计算机集合通过网络互连，实现资源共享和协同工作[10]。虽然对用户来说，集群是一个完整的计算机系统，但实际上集群中的各个组成部分可能分布在不同的地理位置，有着不同的组织结构，而且使用不同的安全策略，同时，程序也分布到不同的机器上运行。为了对集群中的机器进行资源的统一管理和调度，需要一定的策略将这些独立的机器连接起来以便机器间的协同工作，这个策略就是由集群管理系统来提供的。随着科学技术的不断发展，集群技术也在高速发展和广泛应用于各个领域，针对不同的应用场景，衍生出了不同功能和特性的集群类型及其实现方式。

* + 1. 集群类型

根据集群结构和主要功能的不同，我们可以将集群分为负载均衡的集群系统、高可用性的集群系统、高性能的集群系统以及网格计算等几种类型[11]。

（1）负载均衡的集群系统

负载均衡集群的主要目的是让集群能够提供与集群节点个数成正比的负载能力，负载均衡集群非常适合用于提供具有大访问量的Web服务。典型的负载均衡集群有Turbolinux Cluster Server和Linux Virtual Server等。当负载均衡集群处于正常运行状态时，任务调度模块将系统任务按照预先设定的规则分发到后端的一组计算机上进行处理，达到充分利用集群资源和合理分配集群计算能力的目的，避免了部分集群机器任务繁多，需要长时间的排队等待，而另一部分集群机器却长期空闲，无事可做的情况的发生。Linux的虚拟化服务器项目（LVS）提供了最常见的负载均衡的系统软件。

（2）高可用性的集群系统

高可用性集群就是指在集群中的某个或者某些节点在发生故障时，系统会根据相应的规则将失效节点上的任务转到其它节点上执行，也就是说，集群中部分节点的失效并不会影响整个集群系统的正常运行，至于最多能支持几个节点的失效，和集群的存储策略、任务调度策略和故障处理机制等都有关系，不同的系统也各不相同。

（3）高性能的集群系统

高性能的计算机集群是根据算法思想中的分治策略发展而来的，它通过对大量的计算任务进行分解，得到一个个相对较为小且简单的任务，并分配到集群中不同的节点上进行计算，再将计算结果进行整合，实现集群机器间的并行计算，借此来降低任务的执行时间，其大多是应用于科学计算领域。高性能集群比较常用的是HPC，采用了Linuix系统，加上一些开源或者是免费的软件来共同实现并行计算。

（4）网格计算

网格计算，也称为网格集群，是一种和集群计算技术相关的技术。网格集群与传统集群的主要差别在于网格集群是连接的一组计算机虽然相关，但是并不互相信任，在该类型的集群中，单个计算机的运作场景更像是一个用于计算的公共设施。网格计算针对许多独立作业的工作任务进行了优化，使得计算机在计算过程中起作业之间不需要共享数据，因此其主要应用于在独立执行工作的计算机之间进行作业的分配。虽然所有的计算机节点之间共享存储等资源，但是，运算产生的中间结果和其它节点上的任务进展是独立的，不会相互影响。

* + 1. 集群的特征

集群系统的组织结构及其目的决定了其必须拥有如下特征：可用性、可靠性、可拓展性以及CAP理论等。

（1）可用性

系统的可用性（Availability）是指系统在面对各种异常时仍然可以对外提供正常服务的能力，它反映的是系统可以随时被用户使用的特性，即在任何给定的时刻，用户可以使用该系统来正确执行相应的任务[3]。通常情况下，系统的可用性体现的是系统整体代码的质量和容错能力，一般采用系统停止服务的时间与系统正常提供服务的时间的比例来衡量一个系统的可用性。例如，如果一个系统的可用性为，即要求该系统在一年的时间中停止服务的时间最多为分钟。

如果我们使用单机处理任务，当机器的处理器出现问题时，将会导致任务的暂停，要等到处理器被修复时任务才能继续或者重新进行[12]。在集群中，高可用性的引入尽最大可能地限制了此类风险的发生，通常高可用性会涉及到两种不同的机制：快速检测错误机制和快速启动恢复程序机制。快速检测错误机制的关键在于定期检测每个节点的状态，通常此任务会分配给任务管理者，通常就是集群的主控节点（有的也叫做超级节点）；快速启动恢复程序机制是利用复制技术（即将数据复制到多台服务器）和冗余技术（即每个实例会连接多台服务器）来实现，一般来说，在基础设施级别上提供错误管理服务是远远不够的，还必须采用相应的恢复技术来保存一些具有易失性的存储内容。

（2）可靠性

系统的可靠性是指在系统的一个或者多个硬件的软件组成发生故障时，系统仍然能对外提供服务的能力[12]，在集群中，如果有一台参与计算任务的机器发生故障，可以立刻由其他机器替代，完成相应的计算任务，而不会导致请求任务无法完成。可靠性是依赖于数据和软件组件的冗余性能的。以一个电商网站的购物车系统为例，即使一个购物车系统的整个数据中心由于地震等自然灾害而被摧毁，也不会导致系统瘫痪，因为往往会有另外一个备用的数据中心供用户使用。但是，如果像这样通过消除每一个单点故障来实现弹性恢复的服务，随着应用规模的增大，其成本也会不断提高。

系统的可靠性和可用性是不同的概念，但是二者很容易被混淆。可靠性反映的是系统在一段时间内的特性，而可用性反映的则是系统在某一个时刻的特性；高可靠性系统要求能够持续不断的运行一段相当长的时间，而高可用性则是要求系统在一段运行时间内（比如以一年为一个周期）停止时间尽可能少。例如，假设一个系统每个小时有且只有一毫秒的时间会失效，那么，该系统的可用性高达，然而，这个系统仍然是一个高度不可靠的系统；而如果一个系统从来不会崩溃，但是在每年的6月份有两周的时间需要关机维护，那么，这个系统是一个具有高可靠性的系统，但是它不是一个高可用信通，其可用性仅仅为。

（3）可拓展性

系统的可拓展性是指一个系统为了支撑持续增长的任务数量可以不断地进行扩展的能力[12]。由于互联网技术的飞速发展，需要进行存储的数据规模和相应的计算能力等不断扩大，集群规模很可能会超出系统的预期规模，此时，我们就需要在不损失系统性能的情况下完成集群系统的扩展。通常情况下，拓展的方式可以分为横向拓展和纵向拓展，横向拓展是通过增加服务器数量的方式来实现；而纵向拓展则是通过为每一台服务器增加更多的系统资源来实现的。例如，我们现在集群中有100相同的服务器，则理想情况下，每台服务器拥有的资源，需要处理的查询任务，假设现在由于数据的增长，增加了的查询任务，为了保证不影响系统的对外服务，可以通过横向拓展或者纵向拓展来提高系统性能，如果采用横向拓展，就只需要简单地为集群额外增加20台服务器，对并行处理程序几乎没有限制；也可以采用纵向拓展的方式，此时就需要为这100太服务器增加磁盘容量、扩大内存或者是更换处理速度更快的处理器等，但是这种方式对机器的限制性比较大，通常情况下成本也比较高。

（4）CAP理论

CAP理论起源于伯克利加州大学的计算机科学家Eric Brewer在2000年的分布式计算原则研讨会（Symposium on Principles of Distributed Computing, PODC）上提出的一个猜想[12]，这个猜想是，对于一个计算机分布式系统而言，不可能同时满足以下3个条件：

* 一致性（Consistency，C）：所有节点访问同一份最新的数据副本，即在集群中所有的数据备份在相同的时刻具有相同的值。
* 可用性（Availability，A）：对数据更新具有高可用性，即如果集群的一些节点发生故障，集群是否还能正常响应外部用户的读写请求。
* 分区容忍性（Partition tolerance，P）：当集群中的某些节点不能正常联系的时候时，集群仍然能正常提供服务。实际上分区就相当于对通信的时限要求，集群系统如果不能在规定的时限内达成数据一致，就会发生分区，此时，需要就当前操作在可用性和一致性之间做出选择。

后来，这个猜想被证实并规范化，称作CAP定理，如图2-1所示。

由于网络必然存在丢包和延时等问题，因此必须实现分区容忍性，因此在设计分布式系统时就只能在一致性和可用性之间进行二选一。在图2-1中，如果选择了分区容忍性和一致性，那么，如果节点发送故障，操作也必须一致，并能顺利完成，这就要求所有节点之间有很好的连通性，在现实中难以实现，因此最好的办法是将所有数据放到同一个节点上，很显然，这种设计不符合可用性的要求；而如果选择了可用性和分区容忍性，为了保证系统的可用性，数据就必须有不同的节点上有副本，如果此时还要求分区容忍，即产生分区时仍然可以完成操作，那么对一致性就无法保证。



图2-1 CAP理论[12]

* 1. 经典的分布式选举算法

选举算法通常是基于全局优先级的，按照相应的规则和需求，每个节点都会被分配一个预设的优先级，选举的过程就是在所有集群节点中选举出一个所有节点都认可的、具有最高优先级的节点的来担任Master节点。在集群管理系统中引入选举算法，是为了保证集群系统能连续稳定地工作，使集群在Master节点发生故障时，能够快速、有效地选举出新的Master节点，接替原来的Master节点继续执行任务分配、监控集群节点运行状态等功能。如果集群中没有选举算法，一旦集群的Master节点发生故障而无法正常运行时，集群将会陷入紊乱，甚至导致集群瘫痪，造成对外服务的终止和数据损坏。

在分布式系统中，对选举算法的一个基本要求是一致性要求，即对当选为Master节点的节点的选择必须是唯一的，所有节点需要对当选节点达成一致的共识[13]。满足分布式选举算法的要求且仍然继续发展和使用的分布式选举算法主要有以下三种基本算法：邀请选举算法、欺负选举算法和环选举算法。其中，以欺负选举算法的应用和研究最为广泛，论文深入研究了欺负选举算法及其相应的改进算法，并分析了各个算法的优缺点。

* + 1. 欺负选举算法

欺负选举算法[14]是分布式系统的经典选举算法，欺负选举算法是由Garcia-Molin在1982年首先提出的，欺负选举算法具有比较简单的选举流程，满足了选取新的协调节点，保证集群稳定运行的基本要求。该算法其主要思想是：当一个节点发现Master 失效时，就立即向所有优先级比自己高的节点发起选举，依次循环下去，最终选举出一个优先级最高的节点作为新的Master节点。图2-2举例说明了欺负选举算法的整个选举过程，在图2-2中，我们假设集群系统中总共有7个节点，编号从1到7，编号越大，节点的优先级越高，当前Master节点是编号为6的节点。假设节点6失效，节点3是第一个发现节点6失效的节点，则节点3立即发起一次选举，发送选举消息“ELECTION”给优先级比它高的节点4、节点5、节点6和节点7；节点4、节点5和和节点7会恢复一个应答消息，假设此时节点5恢复的消息第一个达到，则有节点5接管选举工作，节点3结束选举工作；节点5接管选举工作后，将发送选举消息“ELECTION”给优先级比它高的节点6和节点7，此时只有节点7会回复应答消息，节点7接管选举工作，节点5结束选举工作；节点7发现没有比它优先级更高的节点，则节点7成为新的Master节点，并广播通知其他节点。



图2-2 欺负算法选举过程

欺负选举算法的具体过程如下：

假设节点K首先发现Master节点已经失效，那么节点K将会发起如下选举过程：

1）节点K从自己维护的优先级表中，选取所有编号比它自身大的节点，并向这些节点发送选举消息；

2）如果节点K没有收到来自其它节点的响应信息，则节点K成为新的Master节点；

3）如果节点K收到来自其它节点的响应信息，则由最早的响应者接替选举工作，重复上述步骤，节点K结束自己的选举任务；

4）最终由当选节点向其他所有存活节点广播自己的Master身份；

5）如果有节点从故障中恢复，且其优先级比当前的Master节点大，则由它发起一次选举。

由上述流程可知，在选举过程中节点间需要进行信息交互，在系统内部会产生相应的通信开销，MohammadRez[15]等人针对欺负选举算法在最坏情况下的通信量和时间开销进行数学分析。假设集群中有个节点，发起选举的节点是编号为的节点，表示由节点发起选举工作时系统产生的通信量，用消息的条数来描述。

1. 节点需要向所有优先级比它高的节点发送选举消息，它发送的消息数量为，则选举过程中所有节点需要发送的的选举消息数量为：

 （2-1）

1. 节点需接收的确认消息数量为，则选举过程中所有节点需要接收的总消息数量为：

 （2-2）

1. 最终选举出的Master节点需要通知集群中其它节点其身份信息时发送的消息数量为

 （2-3）

1. 在一次选举过程中，系统产生的消息总量为：

 （2-4）

由以上理论分析，可以得到如下两个结论：

1）欺负选举算法的通信复杂度过大。在集群节点总数已知的情况下，欺负算法复杂度为。同理，在发起选举的节点的编号已知的情况下，其算法复杂度为。

2）欺负选举算法的通信量波动大。由于选举工作的发起者和接替者都具有较大的随机性，系统中的通信量波动幅度较大。

时间开销也是衡量选举算法的重要指标之一，在文献[16]中，对欺负选举算法的时间开销进行了相应的数学分析[16]。假设集群中有N个节点，其中有k个节点正常工作，则欺负选举算法的时间开销如公式（2-5）所示：

 （2-5）

其中，表示总时间开销，表示两个节点间传递消息所需要的时间，为了简化分析，假设任意两正常节点间传递消息所需时间相同；表示广播Master节点身份所需要的时间。

公式（2-5）表明，当和均为常数时，欺负选举算法的时间复杂度为，集群中存活节点越多，其时间开销就越大，对系统性能的影响也就越大，当集群规模较大时，其选举所用的时间较长，难以忽视，影响系统的可用性。

* + 1. 欺负选举算法的改进算法

根据上面的分析可知，虽然欺负选举算法能够从集群正常运行的节点中选举出性能最好的节点作为Master节点，但是，由于在选举过程中产生的通信量比较大，会占用大量的网络流量，网络延迟较大，严重地影响了系统的性能和工作效率，因此，有很多的学者从各个不同的角度对欺负选举算法进行了优化和改进。

**1、李晓婷的改进算法[17]**

李晓婷在文献[17]中提出的对欺负选举算法的改进算法的主要算法思想是：假设集群中节点P发现Master节点失效，则节点P将发起下面的选举过程：

1. 节点P从自己维护的节点信息全局表中找到所有优先级比自己高的节点，并发送相应的选举信息；
2. 如果节点P没有收到任何响应信息，则节点当选为新的Master节点；
3. 如果节点P收到响应信息，则响应者称为新的Master节点；
4. 最终当选者将当选信息广播给所有节点，表明自己的Master节点身份；
5. 如果一个节点从故障中恢复，则该节点会发起一次选举。

李晓婷改进算法的优点：

该改进算法没有进行循环选举，直接将响应节点作为新的Master节点，通过减少选举的次数使选举过程的通信量降低。

李晓婷改进算法的缺点：

该改进算法忽视了最优解：发起选举过程的节点首先向所有优先级比它高的节点发送选举信息，然后从响应者中选举优先级最高的节点直接作为Master节点，这并不能保证选举出来的节点是当前系统中优先级最高的节点，而仅仅是发起选举的节点所认为的最优节点。

**2、王立军的改进算法[18]**

王立军在文献[18]中对改进对李晓婷提出的改进算法再次做了改进，重新采取了循环选举的策略，以达到选举出具有最高优先级的节点的目的。该改进算法的主要思想是:假设节点P发现Master节点失效，该节点将发起如下选举流程：

1. 节点P从自己维护的节点信息全局表中找到所有优先级比自己高的节点，并发送相应的选举信息；
2. 如果节点P没有收到任何响应信息，则节点当选为新的Master节点；
3. 如果节点P收到响应信息，则由响应节点中优先级最高的节点接替选举工作，节点P结束自己的选举任务；
4. 最终当选者将当选信息广播给集群中的所有节点，表明自己的Master节点身份；
5. 如果一个节点从故障中恢复，则该节点会发起一次选举。

王立军改进算法的优点：

该算法能保证选举出具有最高优先级的节点担任Master节点的角色，且降低了欺负选举算法出现最差选举路径的概率。

王立军改进算法的缺点：

该算法仍然采用了广播的方式将选举信息发送给所有优先级比自己高的节点，本质上只是降低了欺负选举算法出现最差选举路径的概率，并没有真正地降低选举过程的通信复杂度。

**3、吴宁的改进算法[19]**

在文献[19]中，吴宁也提出了一种欺负选举算法的改进算法，该算法的主要思想是：假设节点P发现Master节点失效，该节点会发起下面的选举过程：

1. 节点P从自己维护的节点信息全局表中找到所有优先级比自己高的节点，并向其中任意节点发送选举信息；
2. 如果节点P在规定时间内没有收到响应信息，则节点P当选为Master节点；
3. 如果节点P接收到响应信息，则有响应节点接管选举工作，节点P的选举任务完成；
4. 最终的当选节点将当选信息广播给所有节点，表明自己的Master节点身份；
5. 如果一个节点从故障中恢复，则该节点会发起一次选举。

吴宁改进算法的优点：

该算法意通过一对一地发送选举信息来减少每次选举信息的传递次数，从而达到降低通信量的目的，其通信复杂度从最差情况的到最好情况下的随机变化。

吴宁改进算法的缺点：

该算法使得通信复杂度在到的区间内随机变化，并没有真正意义上的降低算法的通信复杂度。而且，在该算法中，由于选举工作的接替节点具有很大的随机性，导致该改进算法虽然在一定程度上降低了通信复杂度，但是进一步增加了选举算法的时间开销，会导致选举过程的时间变长。

**4、其它改进算法**

MohammadRez在文献[15]中提出的基于最大堆积数的改进欺负算法[15]，降低了用于保存节点优先级的内存大小，但是该算法的思想与欺负选举算法相同，并没有对欺负选举算法的通信复杂度和时间开销进行优化；GholiPour[20]提出的改进算法则是将参与选举的节点进行固定分组，只有在分组里的节点才有参与选举的权利，以减少参与选举的节点数目的方式来达到降低通信复杂度的目的，但是，该算法并没有给出具体的分组依据和标准。

* 1. 心跳检测技术

随着互联网技术的不断发展，集群结构越来越复杂、规模越来越大，各种各样的机器位于不同的环境当中，作为一个整体对外提供服务，集群中机器的运行环境更加复杂，存在着许许多多不确定因素，这些不确定因素会导致节点故障，严重影响系统的正常运行。为了保障数据的完整和安全、系统的快速恢复和系统对外服务的稳定性，容灾机制必不可少。而心跳检测技术正是实现系统容灾机制的前提和基础，它是一种测试网络连接故障的技术[26]，它通过客户端和服务器端的心跳模块之间互相发送心跳信息来检测网络连接或者机器是否正常。

* + 1. 心跳检测模型

根据实现方式的不同，心跳检测技术可以分为PUSH心跳检测模型、PULL心跳检测模型以及二者相结合的混合心跳检测模型[26]。

（1）PULL模型

PULL心跳检测模型是一种问询式的心跳检测模型，在PULL心跳检测模型中，检测节点需要周期性的发送心跳信息给被检测节点，主动询问被检测节点的运行状态，被检测节点在接收到检测节点发送来的心跳信息后，将自己的心跳信息和运行状态发送给检测节点。如果检测节点在时间限制内没有接收到被检测节点的回复信息，就会判定被检测节点失效。PULL心跳检测机制的模型图如图2-3所示。



图2-3 PULL心跳检测模型图

（2）PUSH模型

在PUSH心跳检测模型中，控制流和信息流的当下一致，被检测节点会周期性地、主动地想检测节点发送心跳信息和节点状态信息，如果检测节点在规定的时间内没有接收到来自被检测节点的心跳信息，则判定被检测节点已失效。PUSH心跳检测模型的检测效率较高，但是会出现由于网络延时等因素而导致误判的情况，其模型图如图2-4所示。



图2-4 PUSH心跳检测模型图

（3）PUSH&PULL混合模型

混合心跳检测模型将PUSH心跳检测模型和PULL心跳检测模型相结合，它先使用PUSH心跳检测机制，被检测节点会定期地向检测节点发送心跳信息，如果在规定时间内，检测节点没有接收到被检测节点的心跳信息，则检测节点认为被检测节点有可能已经失效，但是为了避免由于网络延时而导致的误判，检测节点会启动PULL心跳检测机制，主动向被检测节点发送询问信息，询问被检测节点的运行状态，如果被检测节点还在正常运行，就会在接收到检测节点的问询信息后给予相应的回复信息，如果检测节点在规定时间内没接收到被检测节点的回复信息，则判定被检测节点失效。其模型图如图2-5所示。



图2-5 混合心跳检测模型

在传统的心跳检测机制中，PUSH心跳检测模型虽然检测效率较高，但是容易因为网络的延时等原因导致对被检测节点状态的误判；PULL心跳检测模型由于采用的是问询式的检测方式，其检测效率相对较低。

* + 1. HeartBeat项目

HeartBeat是Linux下高可用性项目（High-Availability Linux Project，Linux-HA）的产物，它是一套防止业务主机由于某些不可避免的原因导致宕机引发系统崩溃的高可用性软件[27]。它的工作原理是：对于某一种服务，指定集群中的一个节点作为主节点，另一个节点作为备用节点，正常情况下由主节点对外提供服务，集群中的节点互相发送心跳信息，如果备用节点在规定时限内收不到主节点的心跳信息，就会启功资源转移进程，启动有HeartBeat守护的本机服务并对外提供服务。通过HeartBeat部署的集群结构如图2-6所示。



图2-6 基于HeartBeat的集群结构[9]

在图2-6中，主备节点除了各自节点到外网的连接之外，还需要在主备节点间部署用于传递心跳信息的专用线路，即心跳线，心跳线可以部署一条，也可以部署多条。采用专用的心跳线，而不是直接通过公网传输心跳信息，可以最大限度地避免由于网络问题导致心跳信息的丢失；部署多条心跳线时，只需要从这几条线路中的一条收到心跳信息，备用节点就会认为主节点正常运行，可减少由于网络的意外情况导致的误判。

HeartBeat是Linux-HA的一个成功案例，它提供了高可用集群最基本的功能，其内部结构组成如图2-7所示。



图2-7 HeartBeat内部组织结构[27]

由图2-7可见，HeartBeat的内部组织结构主要分为集群基础设施管理模块（Cluster Infrastructure Management，CIM）、本地资源管理模块（Local Resource Management，LRM）和集群资源管理模块（Cluster Resource Management，CRM）。

集群基础设施管理模块有分为集群成员一致性管理模块（CCM）、节点间通信检测模块（Heartbeat）和集群事件日志服务模块（ha-logd），CCM主要用于管理集群节点成员和负责节点间资源的分配；Heartbeat模块负责检测主次节点的运行状态来判断节点是否失效；ha-logd模块则是用于记录集群运行信息。

本地资源管理模块负责本地资源的启动、停止和监控。该模块有LRM守护进程（lrmd）和节点监控进程（Stonith Daemon）组成，lrmd主要负责节点间通信；Stonith Daemon主要用于监控节点状态。

集群资源管理模块是用于处理集群节点和资源之间的依赖关系并管理节点对资源的使用。该模块由CRM守护进程（crmd）、集群策略引擎（Cluster Policy Engine，CPE）和集群转移引擎组成（Cluster Transition Engine，CTE）组成，其中CPE负责具体实施节点和资源的管理和依赖；CTE则负责监控CRM模块的状态，如果有节点发生故障，则有CTE负责协调另一个节点上的进行资源接管。

* 1. 本章小结

本章从集群出发，介绍了集群的类型和集群的特征等基础理论知识，然后深入到集群的关键技术。选举机制是保证集群稳定运行的前提，本章深入研究欺负选举算法及其部分改进算法，并对这些算法进行理论推导和分析；心跳检测技术是进行故障检测和容灾的前提和基础，本章首先详细介绍了传统的心跳检测模型，然后简要介绍Linux-HA下的HeartBeat的原理和内部组成。

1. 基于节点性能的低通信开销选举算法

在第二章中，对经典的分布式选举算法——欺负选举算法，及其部分改进算法进行了深入的研究和分析，结合集群特点和需求，本章通过在算法中引入让贤和继承的概念，提出了一种基于节点性能的低通信开销选举算法，有效降低了选举过程的通信复杂度。

* 1. 节点性能评估

论文提出的选举机制是基于集群节点性能的，由于集群节点需要进行大量的数值计算工作，同时， Master节点还需要进行集群状态监控以及大量的任务管理和分配，所以主要采用节点的内存大小和CPU核数来作为节点性能的评判标准。论文利用节点能力值（）来描述节点性能，节点能力值的计算方法如公式（3-1）所示。

 （3-1）

式中，表示集群节点能力值；代表集群节点内存的大小，单位为；代表集群节点的CPU核数。论文通过公式（3-1）计算得到的进行节点性能的评估，并将值作为选举过程判断节点优先级的指标，节点的值越大，表示节点的性能越好，代表其全局优先级越高。

在集群中的每个节点上都维护了一张集群节点信息表，节点的值也是节点的属性之一，因此将也存储到该信息表中，而不需要重新建表进行存储。集群节点信息表中存储了节点ID、节点名称、节点状态、节点角色、节点能力值等信息，其设计如表3-1所示。

表3-1 集群节点信息表

|  |  |
| --- | --- |
| NodeID | K |
| NodeName | K |
| Roles | Master/Workers/Inherit\_k |
| Status | OFF/ONLINE/RECOVERY |
| PI | P |
| Reserve | NULL |

在表3-1中，NodeID表示节点序号；NodeName表示节点名称，用NodeID的字符串形式进行命名；Roles表示节点角色，分为Master、Workers以及Inherit\_k，其中k=1,2…，当k=1时，继承优先级最高；Status表示当前时刻节点状态，有OFF、ONLINE和RECOVERY三种；PI表示节点能力值；Reserve为保留字段。

* 1. 让贤与继承机制

论文提出的选举算法主要包含了让贤机制和继承机制两部分。其中，让贤机制在集群初始部署完成或者集群扩容完成时会被触发，选举出优先级最高的节点来担任Master节点，同时根据节点优先级选出相应的继承者节点供继承机制使用；继承机制在原来的Master节点失效时会被触发，由当前正常运行节点中具有最高优先级的继承者节点担任Master节点的角色。

* + 1. 让贤机制

让贤机制在集群初始部署完成时或者集群扩容完成时会被触发，在让贤机制中，除了选举出Master节点，还会选出个相应优先级的继承节点。为了能平稳地部署集群，在集群初始部署阶段，先人为随机指定一台机器作为Master节点，并优先启动这台机器，然后再启动其他机器，当所有机器上线之后，原来的Master节点会启动让贤机制，选举出新的Master节点和所有的继承者节点。

在让贤机制，Master角色的切换是一个原子操作，集群各个节点在完成让贤机制前后的角色转移图如图3-1所示。



图3-1 让贤机制节点角色转移图

在图3-1中，仅以一个新节点加入集群为例进行说明，实际情况下，无论是集群初始部署还是集群扩容，让贤机制都是在所有的新节点都加入集群后才会启动。对于图3-1的一个简单描述是：当新加入的节点的优先级比某个节点高时，该节点的角色就会降低一个级别，否则，维持不变。

让贤机制主要分为三大阶段：第一阶段通过检测是否有新节点加入来确定是否启动让贤机制；第二阶段根据节点能力值选举出Master节点，并让该节点接管Master工作，同时选出相应的继承者节点；第三阶段是由新的Master节点广播通知集群所有正常运行的节点其身份信息。

（1）第一阶段

集群部署阶段和集群扩容阶段的共同特征是集群中有新节点的加入，集群监控系统会通知当前Master节点将会有多少个新节点加入集群，Master节点检测到所有新节点都加入集群后，将让贤标志置为真，并在集群较为空闲时，主动触发让贤机制，选举出新的Master节点和相应的继承者节点。

（2）第二阶段

第二阶段是让贤机制的核心部分。Master节点先从节点信息表中提取所有正常运行节点的节点ID及其对应的能力值，组成key-value的形式，其中key为节点ID，value为节点能力值PI。将节点按照value值的大小进行排序，value值最大的节点即当选为Master节点，如果value值最大的节点仍然是当前Master节点，则结束让贤流程。否则，当前Master节点告知value值最大的节点接管Master工作，接收到该节点的应答后,当前Master节点广播通知所有节点将启动让贤机制。在选出Master的同时，还根据排序结果选出个继承者节点，即value值第二大的节点为第一继承者，value值第三大的节点为第二继承者，以此类推。当有继承者节点失效时，继承者根据value排序结果下移并由Master节点更新信息表且同步到所有节点，当其从失效状态恢复时，其继承者身份也随之恢复，同样地，由Master节点更新信息表且同步到所有节点。但是，当原来的Master节点从失效中恢复时，为了保障集群的稳定性，并不会立即恢复其Master身份，而是根据其节点能力值将其加入到继承者优先级队列中，通常情况下，该节点会成为第一继承者。

在集群初始部署阶段或者是在集群扩容时，当所有节点都加入集群后，系统才会启动让贤机制，这保证了在同一时刻，无论节点状态是否有效，集群中所有节点在完成排序的value队列中的位置唯一。而且一般情况下各个节点的能力值不会发生改变，因此只有在集群初始部署阶段和集群扩容时才需要进行排序操作，当节点失效或者恢复时只需要变更节点信息表中的节点状态。

（3）第三阶段

当集群中某个节点接收到来自当前Master节点的接管信息时，该节点以Master身份运行，同时响应原来的Master节点，原来的Master节点收到应答后以普通节点身份运行。新的Master节点更新节点信息表，并通过广播告诉所有有效节点其Master身份，同时向其它节点同步信息表。

* + 1. 继承机制

在集群中Master节点需要负责监控集群其他节点的运行状态和进行大量任务分配调度，一旦Master节点失效，且没有其它节点及时接管工作，集群将会瘫痪。继承机制的引入就是为了在Master节点失效后，及时地选举出一个新的Master节点来管理和监控集群状态。

继承机制的主要思想是：当节点发现Mater失效时，立即到本地信息表中查询自己的角色信息，如果该节点是第一继承者则发起继承，否则就等待第一继承人的继承信息，如果在在规定时间内第一继承者没有响应，则进行第二继承者的继承工作，以此类推。

由于各节点间的信息表可能存在不同步的情况，会存在有多个节点都认为自己是第一继承者的情况，此时需要进行冲突处理。冲突处理机制如下：当节点检测到冲突时，则该节点将本地信息表和冲突节点信息表的时间戳进行比较，如果自身的时间戳较旧，则该节点主动放弃竞争，否则继续进行继承流程。

当节点发现Master节点失效时，继承机制的主要流程如下：

（1）节点查询本地集群信息表如果是第一继承者，则发起继承，否则进行等待；

（2）如果发生冲突，则根据时间戳信息进行冲突处理，时间戳较老的主动放弃竞争；

（3）如果在规定时间内没有第一继承者发出继承信息，则进行第二继承者继承流程，与上述步骤一样；

（4）如果第一继承者收到的响应消息数量与当前存活节点数量一致，则顺利继承，成为新的Master节点，接管Master工作。

（5）新的Master节点向集群广播自己的身份信息。

在集群执行继承机制前后，集群信息表会发生较大的变更，如图3-2所示。



图3-2 执行继承机制前后信息表变化情况

在图3-2中，由于原来的Master节点发生故障，集群执行了继承机制。在继承机制执行完成之后，原来的Master节点的角色转变为Workers节点，且其运行状态变为OFF；第一优先级继承者顺利进行继承，角色转变为Master节点；其余的继承者优先级依次提升一个级别。

* 1. 性能分析与仿真

为了评估基于节点性能的低通信开销选举机制的性能，需要进行仿真验证，并和欺负选举算法进行比较分析。本节先从理论层面对让贤机制和继承机制两大机制分别进行了推导和分析，得到这两大机制的通信复杂度，在进行理论推导时，假设选举过程中集群每个节点发出的每一条消息的数据大小相等，以选举过程中的产生的消息条数作为衡量算法通信复杂度的指标。然后再根据得到的通信复杂度进行仿真分析。

* + 1. 让贤机制通信复杂度

在进行理论分析之前，先做如下几点假设：

（1）假设集群中的节点总数量为，为大于等于2的正整数，在集群中不存在失效节点，而且集群各节点之间网络连接正常；

（2）假设在选举过程中，每个节点发出的选举消息的数据大小相等；

（3）假设当前集群的Master节点为节点，则让贤机制将由节点发起，为大于零且小于等于的正整数。

则进行一次让贤所需的通信量如下所述，其中，用表示集群中由Master节点发起的让贤机制所产生的消息条数。

（1）当前Master节点在确定集群中有比自己更优的节点之后，需要执行让贤过程来进行角色更换，首先需要通知集群中所有其它节点即将进行让贤过程，然后将会与集群中其它节点暂时中断连接，要求集群中其他节点不要向自己发送信息，等待新的Master节点的通知，使让贤过程处于“原子”状态，因此Master节点首先需要发出的消息数量：

 （3-2）

（2）当前Master在进入让贤过程时，需要与选举出的新的Master节点取得联系并达成角色交换的协议，因此原来的Master节点需要单独向选举出的新的Master节点发送接替信息并取得回复，在这个过程中，产生的消息数量为：

 （3-3）

（3）在角色接替完成之后，新的Master节点需要通知集群中所有其他节点自己的身份信息并建立连接，因此新的Master节点需要向集群中所有其他节点发出一条通知消息，其消息数量为：

 （3-4）

（4）新的Master节点发出通知消息后，由于需要取得集群中其它节点的认可，并建立连接通道，因此要求其它节点回复相应的信息，其需要接收的消息数量为：

 （3-5）

（5）由节点发起的让贤流程产生的消息总数：

 （3-6）

从公式（3-6）可以看出，让贤机制的通信复杂度为，而且，在不考虑节点失效的情况下，同通信量和集群节点数量呈确定的线性关系，当集群节点数量一定时，让贤机制在选举过程中产生的通信量不会发生随机性的波动。

* + 1. 继承机制通信复杂度

同样的，在对继承机制进行分析之前，先做如下几点假设：

（1）假设集群中的节点总数量为，为大于等于2的正整数，在集群中不存在失效节点，而且集群各节点之间网络连接正常；

（2）假设在选举过程中，每个节点发出的选举消息的数据大小相等；

（3）假设当前集群的Master节点为节点，当当前的Master节点失效时，继承机制由节点发起，可能有多个，其中，为整数且；为整数且。

（4）假设，Master失效时集群中存在个冲突的第一继承者。

则继承机制中最坏情况下所需的通信量如下所述，用来描述由节点发起继承时，选举过程中产生的消息数量如下所述。

（1）第一优先级的继承者节点在发现原来的Master节点失效之后，该节点需要通知集群中所有其它节点自己是第一优先级继承者，即将接管Master节点的角色，则第一优先级继承者节点需要发出的消息数量为：

 （3-7）

（2）第一优先级继承者节点在发出继承消息后，需要得到集群中其它所有节点的认可，确保没有继承冲突发生，或者是发生的继承冲突已解决，并与其它节点建立连接，则第一优先级继承者节点接收的消息数量为：

 （3-8）

（3）为分析方便，忽略时间戳最新的第一优先级继承者节点不会回复其他基础者节点信息的情况，则继承过程中产生的消息总数量为：

 （3-9）

由公式（3-9）可知，当第一继承者节点的冲突数量已知的情况下，继承机制的通信复杂度为。

* + 1. 仿真结果

集群节点数量用进行描述，并假设在选举过程中除了原来的Master节点外，集群中不会有节点失效。当分别取、时，对欺负选举算法、让贤机制和继承机制分别进行次的仿真，以三者在选举过程中产生的消息条数作为通信开销的衡量指标，仿真结果如图3-3和图3-4所示。



图3-3算法通信量（nodeNum=10）



图3-4算法通信量（nodeNum=20）

从图3-3和图3-4可以看出，在大多数情况下，让贤机制和继承机制的通信量远低于欺负算法。而且，欺负选算法举由于选举过程的发起者具有较大的随机性，在50次实验当中，其产生的消息数量呈现出剧烈的波动，以集群节点数量为20台的情况为例，欺负算法产生的消息数量少的只有二十多条，多的则达到了400条以上；而让贤机制和继承机制则不会存在这方面的问题，其通信量十分平稳。

表3-2给出了让贤机制、继承机制和欺负选举算法在不同集群规模下通信量的对比，表中数据是在不同集群节点数量的情况下，各取50次仿真结果的平均值所得。

表3-2 算法平均通信量/条

|  |  |  |  |
| --- | --- | --- | --- |
| 节点数 | 20 | 30 | 50 |
| 欺负算法 | 195 | 325 | 895 |
| 让贤机制 | 59 | 89 | 149 |
| 继承机制 | 76 | 116 | 196 |
| 让贤机制算法优化率 | 69.7% | 72.6% | 83.4% |
| 继承机制算法优化率 | 61% | 64.3% | 78.1% |

从表3-2可以看出当集群节点数在20个以上时，让贤机制和基础机制在选举过程中产生的平均通信量都比欺负选举算法降低了以上，且随着集群节点数的增加其优化率有进一步提高的趋势，具体如图3-5所示。



图3-5 不同集群规模下的算法通信量

从图3-5中可以看到三种算法在选举过程中产生的通信量随着集群规模扩大的变化情况。整体而言，三种算法都是随着集群规模的扩大，通信量也随着增加，但是从图中可看出，让贤机制和继承机制的曲线斜率远远小于欺负算法的曲线斜率，即二者产生的消息量的增加幅度远小于欺负算法的增加幅度，且随着集群规模的增大，让贤机制和继承机制的优势更加明显。

* 1. 本章小结

本章主要围绕着基于节点性能的低通信开销选举算法展开，首先讨论了节点性能的评价指标，然后深入分析算法的主要思想，最后给出性能分析和仿真结果。

该选举算法主要针对欺负选举算法通信复杂大，且由于选举发起者的随机性引起的通信量波动而提出的，以节点性能为优先级指标，通过让贤和继承两大机制来实现，仿真结果表明，与欺负算法相比，该算法在集群节点数达到20台以上时，通信量能降低60%以上，且随着集群规模的扩大，优势越明显；此外，该算法的在选举过程中通信量十分平稳，不会出现波动。

1. 结合同步信息的心跳检测机制

在论文第二章中介绍了传统的心跳检测模型，其中，PUSH模型由于存在容易因为网络延时导致误判的问题，虽然检测效率高，但是实用性不高；PULL模型采取的问询式的检测方式，其检测效率相对较低；混合检测模型也只能实现单向检测，而在集群中，不仅仅Master节点要监控普通节点的运行状态，普通节点也需要知道Master节点是否还在正常运行。论文在混合检测模型的基础之上，结合集群系统的具体需求，对其进行改进，采用了基于混合检测模型的、双向的心跳检测机制。同时，为了提高网络传输效率，将集群信息的同步也结合到心跳检测机制当中，形成一套结合同步信息的双向心跳检测机制。

* 1. 双向心跳检测机制

由于环境等诸多不确定性因素的存在，集群中的任何节点在任何时刻都有可能发生故障，系统需要能及时发现那些节点发生故障，并采取相应的措施进行应对。因此，在集群中，不仅Master节点需要检测普通节点的运行状态，普通节点也需要知道主控节点是否还在正常运行。如果Master节点检测到普通节点发生故障，就需要进行信息表更新和任务转移等操作；而如果普通节点检测到Master节点失效，就会发起选举，选出一个新的Master节点来接管工作。传统的单向心跳检测模型不能满足上述要求，论文在PUSH&PULL混合心跳检测模型的基础上进行改进，采用了如图4-1所示的双向心跳检测模型。

在图4-1中，上半部分由Master节点担任检测节点，集群中的普通节点作为被检测节点，其心跳检测流程如下：

（1）首先，集群中的普通节点会周期性地push心跳信息给Master节点；

（2）如果在限定的时间内，Master节点没有接收到来自某个普通节点的心跳信息，则判定该节点心跳超时；

（3）为了进一步确认该超时是由于网络延时导致，还是由于节点失效引起的，Master节点会主动发送心跳问询信息；

（4）该节点在接收到此问询信息后，就会立即重新发送一次心跳信息给Master节点；

（5）如果Master在规定时间内没有接收到来自该节点的心跳信息，则判定该节点失效，采取相应的处理措施。

在图4-1中，下班部分与上半部分是对称的，Master节点作为被检测节点，普通节点作为检测节点，其心跳检测流程如下所述：

（1）首先，集群中的Master节点会周期性地push心跳信息给集群中其它节点；

（2）如果在限定的时间内，集群的某个节点没有接收到来自Master节点的心跳信息，则判定Master节点心跳超时；

（3）为了进一步确认该超时是由于网络延时导致，还是由于节点失效引起的，该节点会主动发送心跳问询信息；

（4）Master节点在接收到此问询信息后，就会马上重新发送一次心跳信息给该节点；

（5）如果该节点在规定时间内没有接收到来自Master节点的心跳信息，则判定Master节点失效，执行选举等操作流程。



图4-1 双向心跳检测模型

双向心跳检测模型在PUSH&PULL混合检测模型的基础上，使用了双向检测机制，满足了集群普通节点需要知道Master节点是否正常运行的需求，在保证较高的检测效率时，通过补充PULL的机制降低了误判情况发生的可能性。

* 1. 同步信息

为了降低成本，集群一般是由普通的小型机器组成的计算机系统，节点机器的价格比较便宜，但是，机器发生故障的概率也相对较高。一旦机器发生故障，尤其是出现硬盘损坏等情况，会造成机器上本地信息的丢失，如果没有对这些信息进行冗余备份，将会造成大量数据的丢失且不可恢复，带来难以预估的后果。一般而言，信息同步包括集群信息同步和存储数据同步两大部分，集群信息同步是指将集群的节点状态信息和集群存储相关的元数据等信息同步到集群的其他节点上进行冗余备份，存储数据同步是指在进行新数据存储时，为了提高数据的安全性，对数据进行冗余备份，一般也是备份到集群其它节点上，且业界一般采用的是3备份策略。本文中提到的信息同步特指集群信息同步这一部分，不涉及具体的存储数据的同步。

集群的Master节点需要监控集群其他节点的运行状态并将节点状态变更记录到集群节点信息表中；此外，Master节点还需要管理集群存储相关的元数据，维护存储相关的信息表。一旦Master节点发生故障，会导致相应的信息表数据丢失，此时，如果没有相应的保护措施，集群系统将会瘫痪，需要重新组建集群，而且，大量的数据由于失去了相应的存储元数据而无法进行恢复，带来巨大的经济损失。因此，需要有信息同步机制将集群信息同步到其它的集群节点进行备份冗余，保证在集群的Master节点发生故障后集群中仍然存在相应的元数据，这是保障集群可靠性和数据安全性的前提之一。

论文以结构化数据的列式数据库存储为集群背景，Master节点维护的信息表有集群节点信息表（InfoNode Table）、字段信息表（DFS\_Index\_Table，DFS\_IT）和数据块存储信息表（DFS\_Allocation\_Table，DFS\_AT）。为保证在Master节点发生故障时集群数据的安全性，需要将这三张表及时同步到集群其它节点上。

InfoNode Table的表结构如表3-1所示，在第三章已有介绍，此处不再赘述。

DFS\_IT的表结构如表4-1所示。

表4-1 DFS\_IT表结构

|  |  |
| --- | --- |
| bDataBaseID | K |
| bTableID | M |
| bFieldID | N |
| RecordCount | H |

在表4-1中，bDataBaseID表示数据所属数据库的数据库编号，定义为unsigned int类型，K取正整数；bTableID表示字段所属的数据表，定义为unsigned int类型，M取正整数；bFieldID表示数据所属字段的字段编号，定义为unsigned int类型，N取正整数；RecordCount记录某一字段数据的条数，同样定义为unsigned int类型，H取正整数。对于DFS\_IT表而言，每一张表都对应于一个字段的存储属性，例如，假设字段“温度”的变化为2，当K等于1，M等于2，N等于2，H等于10000时，就代表着此处存储的数据代表温度的值，它属于数据库编号为1的数据库中编号为2的数据表，该数据库中，当前总共有10000条相应的温度记录。

DFS\_AT的表结构如表4-2所示。

表4-2 DFS\_AT表结构

|  |  |
| --- | --- |
| wNodeID | K |
| DBIndex | M |
| wFlag | 0/1 |
| qwOffest | N |

在表4-2中，wNodeID表示数据存储的机器的节点编号，定义为unsigned int类型，K取正整数；DBIndex是数据存储的数据块的索引，同样定义为unsigned int类型，M去正整数；wFlag标识定义为bool类型，表明该字段在该节点上的最后一个数据块（DataBlock，DB）是否存满，0代表该数据快未存满，下次存储应该先将此数据块填满，1代表该数据已存满，下次存储需要寻找新的数据块进行存储；qwOffest定义为unsigned int类型，N取正整数，表示某些数据在该数据块中从哪个位置开始存储。每一张DFS\_AT表对应于DFS\_IT表中的一种字段，记录着对应字段的具体存储信息。

* 1. 结合同步信息的双向心跳检测机制

由于心跳信息本身比较简短，甚至一个字节就完全足够用来表示节点的运行状态，在传输过程中需要对其进行补零操作以满足网络传输的要求，传输效率较为低下，在一定程度上造成网络资源的浪费。考虑到集群系统中存在信息同步的需求，可以将同步数据和心跳信息进行组合，然后再统一进行发送，既实现了信息的同步，又能提高网络传输效率。结合同步信息的双向心跳检测机制的系统框图如图4-2所示。



图4-2 结合同步信息的双向心跳检测机制

在图4-2中，对于集群的普通节点而言，需要周期性地向Master节点发送心跳信号，由于集群的普通节点没有向Master节点同步信息的必要，但是却需要向Master汇报其本地的资源负载情况，这个资源负载情况是会随时发生变化的，因此，当集群普通节点需要发送心跳信号时，该节点会读取当时机器的本地资源负载情况，和心跳信号进行数据整合后发送给Master节点；而对于Master节点而言，需要将一个心跳周期内集群节点信息表、DFS\_IT表和DFS\_AT表的变化内容同步到集群的其它节点上，因此，每当这三张信息表发生变更时，Master节点会将相应的变更信息先写入缓冲区，当需要发送心跳信息时，再从缓冲区中读取变更数据，和心跳信号进行整合后再发送给普通节点。

在结合同步信息的双向心跳检测机制中，Master节点和普通节点分别在PUSH心跳信息时携带相应的信息，如表4-3所示。

表4-3 心跳信息组成

|  |  |
| --- | --- |
| 普通节点 | 主控节点 |
| KeepAlive | KeepAlive |
| hardWareUsage | flagInfoNode、flagIT、flagAT |
| CPUUsage | changeInfoNode |
| memoryUsage | changeIT |
|  | changeAT |

对于集群的普通节点而言，心跳信息中携带了节点存活信号和节点的本地资源负载情况，其中KeepAlive信号定义为bool类型，置1时表示节点处于正常运行的状态；hardWareUsage表示节点的硬盘利用率Master节点通过这个参数可以知道某个集群节点的硬盘是否还有充足的空闲空间用于存储数据；CPUUsage 表示集群节点的CPU利用率，memoryUsage则表示集群节点的内存利用率，Master通过这两个参数可以知道当前该节点的算力使用情况，以及还有多少计算资源可供利用。hardWareUsage、CPUUsage和memoryUsage三者都采用double型的数据类型进行表示。

对于Master节点而言，心跳信息主要携带了节点存活信号和集群信息表的变更信息，其中KeepAlive信号和普通节点的KeepAlive信号一样，定义为bool类型，置1时，表示Master节点处于正常运行的状态。flagInfoNode、flagIT和flagAT三者也都是定义为bool类型的数据，其中flagInfoNode标识集群节点信息表是否发生变更，置1时代表该信息表有信息发生改变，置0代表该表没有发生变动；flagIT标识DFS\_IT表单内容是否发生改变，置1时代表该表单有信息发生改变，置0代表该表单没有发生变动；而flagAT标识DFS\_AT表单内容是否发生改变，置1时代表该表单有信息发生改变，置0代表该表单没有发生变动。changeInfoNode、changeIT和changeAT三者均为字符串类型的数据，其内容分别为集群节点信息表、DFS\_IT表和DFS\_AT表的变更内容，如果某个表单内有信息发生改变，则该表单对应的字符串为空值。

* 1. 性能分析与仿真

结合同步信息的双向心跳检测机制在保有和PUSH&PULL混合检测模型相同的检测效率的同时，提高了系统的传输效率。接下来就通信开销和传输效率两个方面进行分析，并给出相应的仿真结果。

* + 1. 通信开销

论文以完成一次心跳检测动作所需要的时间作为心跳检测机制通信开销的衡量指标，为了分析方便，论文假设在不同的心跳检测模型下，单向传输信息的时间相同，即单向传输的通信开销相同，将其定义为，并称作单位通信开销。上文提出，PUSH心跳检测模型容易由于网络延时等原因而导致误判的情况发生，其本身实用性较低，论文只将PULL心跳检测模型作为对比对象。

当采用PULL心跳检测模型时，由于采取的是问询式的检测方法，所以检测节点需要先向被检测节点发送询问信息，被检测节点在接收到检测节点发来的询问信息之后，向检测节点发送相应的心跳信息，被检测节点接收到检测节点的回复信息后便完成了这一次的心跳检测，其通信开销为：

 （4-1）

在论文的结合同步信息的双向心跳检测模型中，集群的Master节点和普通节点都有着心跳检测信息的发送模块和接收模块，两个方向的心跳检测之间不会相互影响，因此只需要计算单方向心跳检测的通信开销即可，其通信开销与PUSH&PULL混合检测模型相同。在双向心跳检测模型中，首先采用PUSH机制，被检测节点会先主动发送心跳信息，如果检测节点在规定时间内没有接收到来自被检测节点的心跳信息，检测节点就会启用PULL机制，向被检测节点发送问询信息，在接收到被检测节点回复信息后完成此次心跳检测，如果在规定时间内没有接收到被检测节点的回复信息，则判断被检测节点失效，结束此次心跳检测。由于PUSH心跳检测模型只需要单向发送心跳信号，其通信开销为：

 （4-2）

假设由于网络延时等原因导致PUSH心跳信息时发生超时或者丢失的概率为，，则完整的双向心跳检测模型的平均通信开销约为：

 （4-3）

根据公式（4-1）和公式（4-3），假设等于0.03，则PULL心跳检测模型和双向心跳检测模型的通信开销如图4-3所示。



图4-3 心跳检测模型通信开销仿真（*p=0.03*）

图4-3反映了当PUSH超时失效概率确定时，双向心跳检测模型的通信开销比PULL心跳检测模型的通信开销有所优化，当等于时，其通信开销约为PULL心跳检测模型通信开销的一半。

很显然，双向心跳检测模型的通信开销与PUSH超时失效概率息息相关，图4-4和表4-4给出了在单位通信开销固定时，双向心跳检测模型和PULL心跳检测模型的的通信开销情况，此处等于毫秒。



图4-4 心跳检测模型通信开销仿真（*k=10ms*）

图4-4反映了在单位通信开销固定，且取值10毫秒时，双向心跳检测模型和PULL心跳检测模型的通信开销随着PUSH超时失效概率的值的改变而变化的情况。从图4-4中可以看出，当时，双向心跳检测模型的通信开销比PULL心跳检测模型要来得低，且的值越小，其优化率越高，具体如表4-4所示，当等于0时，优化率为50%，没增加0.1，优化率就降低10%，直到等0.5时，优化率为0；当时，双向心跳检测模型和PULL心跳检测模型的通信开销相同；而当时，双向心跳检测模型的通信开销要比PULL心跳检测模型来得高，具体如表4-4所示。因此，双向心跳检测模型适用于网络情况较好的集群，考虑到大部分集群采用的是专网通信，网络情况相对良好，值也相对较小，因此双向心跳检测模型也有较大的实用价值。

表4-4 *p*取不同值时心跳检测模型通信开销（*k=10ms*）

|  |  |  |  |
| --- | --- | --- | --- |
|  | 双向心跳检测模型 | PULL心跳检测模型 | 优化率 |
| 0 | 10 | 20 | 50% |
| 0.1 | 12 | 20 | 40% |
| 0.2 | 14 | 20 | 30% |
| 0.3 | 16 | 20 | 20% |
| 0.4 | 18 | 20 | 10% |
| 0.5 | 20 | 20 | 0 |
| 0.6 | 22 | 20 | — |
| 0.7 | 24 | 20 | — |
| 0.8 | 26 | 20 | — |
| 0.9 | 28 | 20 | — |
| 1 | 30 | 20 | — |

* + 1. 网络传输效率

在网络传输当中，对数据包大小有相应的规定，以传输层的TCP协议为例，数据包大小范围是60字节到1500字节，当数据大小大于1500字节时会进行拆分，当数据包大小小于60字节时会进行补零。如果传输的是小数据，有效数据所占比例小，传输效率低下。如果心跳检测中只传输keepAlive信息，只有一字节，其传输效率为：

 （4-3）

将同步信息与心跳信号相结合之后，同步信息中只记录表的变更信息，包括变更的位置，以及变更后的数据，根据上述的表结构设计，InfoNode Table中每条变更记录的大小可设定为16字节，DFS\_IT中每条变更记录的大小可设定为11字节，DFS\_AT中每条变更记录的大小可设定为16字节。假设在一个周期内InfoNode Table中的变更记录数为，DFS\_IT中的变更记录数为，DFS\_AT中的变更记录数为，则一个心跳周期内总变更记录的数据大小为：

 （4-3）

在采用TCP/IP协议进行传输的过程中，如果只考虑传输层和网络层的数据拆分和封装，拆分后的每个数据包都包含了20字节的IP头和20字节的TCP头，每块数据中的有效数据最大为1460字节，当传入的数据大于1460字节时，需要进行拆分，设传入的数据被拆分为个数据快，值由公式（4-5）确定。

 （4-5）

其网络传输效率为有效传输数据和总传输数据的比值，如公式（4-6）所示。

 （4-6）

为分析方便，公式（4-6）中忽略了当拆分后数据库大于1时，最后一个数据块的数据量极少的情况（即一个数据块的数据量小于20字节时），而是将其当作大于20字节的情况处理，对分析结果影响很小，可以看作等同效果。

根据公式（4-6），可以得到如图4-5所示的结合同步信息的双向心跳检测机制在S大小不同的情况下的传输效率。

图4-5网络传输效率

从图4-5可以看出，当变更信息的数据量为0时，与只传输心跳信号等同，传输效率仅为1.67%，随着数据的增加，传输效率逐步提高，最后在一个较小的范围内波动，当数据量大于500字节时，传输效率均达到90%以上。从图中可以看出，数据量较大时，其传输效率存在上下波动，这是由于随着数据的增加，会周期性的出现新的数据分片，当新的数据分片中数据量较少时，其传输效率会受到一定的影响。整体而言，结合同步信息的心跳检测机制的传输效率比仅传输心跳信号时有了大幅度的提高，当变更信息数据量在500字节以上时，传输效率能提高90个百分点以上。

* 1. 本章小结

本章主要围绕结合同步信息的双向心跳检测机制展开。首先介绍了双向心跳检测机制模型及其必要性，双向心跳检测模型是基于PUSH&PULL混合检测模型的，在该模型中集群的Master节点和普通节点同时作为检测节点和被检测节点，两个检测方向以对称的形式进行，不会相互影响；之后描述了集群信息同步的必要性及其同步内容，同步的内容主要针对集群的三张全局表展开，同步时需要传输的同步信息包含这三张全局表的所有变更信息，包括变更的位置，以及变更后的数据等；然后给出了结合同步信息的心跳检测机制的设计；最后对该机制在通信开销和传输效率两个方面进行分析和仿真，结果表明该机制在通信开销和传输效率上均有所优化。

1. 集群管理系统分析与设计
   1. 应用场景

当下，集群技术广泛应用于各行各业之中，包括金融、互联网和科研机构等，其应用场景也各不相同，有的将集群技术应用于数据的存储和查询，有的则是将其应用于数据挖掘和数据分析处理，总的来说，其应用场景可以分为联机事务处理（On-Line Transaction Processing，OLAP）和联机分析处理（On-Line Analytical Processing，OLTP）两大类别。OLTP主要是执行基本的、日常的事务处理操作，比如数据库数据的增添、删除、查找和改动等，一个常见的、典型的事务是在银行的一笔交易记录的处理，OLTP系统一般要求较高的实时性，其数据量不是很大，且需要支持高并发和要求满足ACID原则；而OLAP则是主要应用于数据仓库，需要支持复杂的计算分析操作，侧重于高层决策支持，并且需要提供直观易懂的结果，动态的报表系统就是一个典型的OLAP应用，OLAP系统一般来说对实时性的要求不高，但是其数据量较大，OLAP的查询一般是动态的，因为可能需要将许多数据进行统计、计算和分析后才能给出用户想要的结果。

本课题以OLAP系统应用为背景，针对海量结构化数据的存储和计算分析，通过对心跳检测技术、信息同步技术和选举机制等关键技术的深入研究，设计了一个安全可靠的集群管理系统，该系统能及时检测节点故障并及时进行容灾处理，具有较高的可靠性，能保证集群数据的安全。

* 1. 集群管理系统拓扑结构

在集群的构建方面主要有两种系统模型，它们分别是中心化模型和无中心化模型。中心化模型，也称为集中式模型，是一种传统的通信网模式，它以一个机器作为中心机器节点，并且由该中心机器节点统一控制集群中的其他机器节点，中心机器节点和集群中其他所有的机器节点通过一条物理线路进行连接，并且给所有的机器节点指定一个唯一的编号，如图5-1所示。在中心化模型中，机器节点之间的通信都需要说明目的机器节点的编号，并且需要先将数据发送到中心机器节点，在由中心机器节点将数据转发到目的节点。



图5-1 中心化网络模型

中心化模型的结构十分简单，只需要确保中心机器节点和集群中其他机器节点之间的连通是可靠的就能正常运行。但是，事实并非这么简单。首先，中心化模型的网络拓扑的健壮性较差，如果集群的中心节点由于某种自然原因或者非自然原因而损坏，那么，整个集群就会陷入瘫痪，尽管有双击热备等技术的引入，但是如何保证不会有主备机器同时损坏的情况发生也是一个难以解决的问题；其次，直接在中心机器节点和集群的其它节点建立一条可靠的物理通道并没有想象中那么容易，尤其是在机器相隔较远、线路较长时，成本和可行性都会存在问题；再者，在中心化模型中，由于集群中其他机器节点之间的数据传输需要经过中心机器节点进行转发，一旦集群规模较大，交互信息大幅增加，或者是集群内的机器之间需要进行大量数据的转移、备份时，系统的网络中会有大量的数据需要传输，不仅仅对网络性能提出更大的挑战，而且对中心机器节点的性能要求也非常之高，尤其是在网络IO方面的要求，因为中心机器节点其实是担任着类似于路由器的功能，所有数据都需要经过该机器，中心机器节点容易导致性能瓶颈。

中心化模型的主要问题在于集群所有节点都由中心机器节点进行控制，数据的传输都需要经过中心机器节点。如果将这个中心机器节点去掉，并通过网络协议，如TCP/IP等，使集群中的机器实现互联，就能建立起一套健壮的、稳定的网络了，这就是去中心化的分布式系统网络，如图5-2所示。



图5-2无中心网络模型

在无中心模型中，集群中的每台机器不需要都有直接的物理通道便能实现通信，且不会由于存在中心机器节点而导致系统的性能瓶颈等问题。本课题的集群拓扑结构是基于无中心模型的思想进行设计的，集群系统的逻辑拓扑和物理拓扑分别如图5-3和图5-4所示。



图5-3 集群逻辑拓扑

图5-4 集群物理拓扑

虽然集群采用的设计思想是基于无中心模型的，但是由于需要对集群机器节点的运行状态进行监控、管理集群的数据存储以及进行任务的调度和分配等，所以集群中需要一个Master节点来承担这些工作。Master节点除了额外承担这些工作外，其余的功能和普通节点是对等的，也就是说，Master节点仍然需要执行数据的存储和计算等任务。

* 1. 系统总体功能规划

本课题对集群管理系统的设计主要围绕系统的安全性和可靠性进行，系统的主要功能是对集群节点进行监控和管理以及检测集群故障并进行相应的处理，系统的关键功能模块主要有Master 节点选举模块、心跳检测和信息表同步模块、节点上线检测模块以及故障处理模块等，其中，节点上线检测模块又可以进一步细分为新节点上线处理模块和故障节点恢复上线模块；节点故障处理机制又可以细分为Master节点故障处理机制和普通节点故障处理机制，节点故障处理机制并不是一个单独的模块，它是由集群其他模块协同工作，并配合一些外部处理模块的功能，对节点故障进行处理，达到维护集群稳定运行和数据安全的目的。本课题设计的集群管理系统的关键功能模块如图5-5所示。

图5-5 集群管理系统关键功能模块

课题以图5-5所示的关键功能模块为基础进行集群管理系统的设计，系统的程序框图如图5-6所示。



图5-6 系统程序框图

在图5-6中，集群中的机器角色分为Master节点和普通节点，在Master节点和普通节点上有对应的功能模块，相同的模块在Master节点和普通节点上虽然有着相同的功能，但是二者在具体实现上的细节会有所区别。

* + 1. 选举模块

在论文第三章已经给出了系统采用的选举算法的主要思想并对其性能进行仿真分析，该选举算法分为让贤机制和继承机制，本小节主要对着两大机制的实现进行介绍和分析，这两大机制都在选举模块中进行实现，其中Master节点对应的是让贤机制，而普通节点对于的则是继承机制。

（1）让贤机制

当系统中有新的节点加入时，等待所有新节点全部加入完成，集群的Master节点会将系统中的让贤标识置位真，当系统较为空闲时，让贤机制就会被触发。在让贤机制中，集群中的机器节点可以分为以下三种角色：原来的Master节点、选举出的新的Master节点以及其他的普通节点。图5-7为让贤机制的流程图。



图5-7 让贤机制流程图

图5-7反映了让贤机制的整体程序设计流程，根据第三章选举算法的思想进行让贤机制流程的设计和实现，让贤机制首先由Master节点发起，在角色交换之前，原来的Master需要检测继承节点上信息表是否是最新的，即是否与其同步，这是为了保证集群信息元数据的安全性；在进行让贤操作时需要中断与集群其他无关节点的交互，保证让贤是一个原子操作，不会受到外界的干扰；在完成角色切换后，新的Master节点需要对信息表进行更新，主要是更新节点的角色信息，保证集群信息的正确，并将其同步到集群其他节点。

（2）继承机制

当集群的Master节点由于某种原因而发生故障后，其它节点无法接收到其心跳信息，此时集群的其它节点就会判断Master节点失效，并发起继承机制，选举出新的Master节点来接替工作，维持系统的正常运行。继承机制的处理流程如图5-8所示。



图5-8 继承机制流程图

图5-8反映了继承机制的整体程序设计流程，继承机制是在集群Master节点失效后，由具有最高继承优先级的节点发起，在完成继承后，新的Master节点需要进行信息表的更新操作，主要是更改自身身份信息为Master，以及将原来失效的Master节点更改身份信息并将其状态更改为OFF状态，然后将信息同步到集群其他节点。

由于集群在执行继承机制时，可能存在信息表不完全同步而导致集群中有多个机器节点认为自己是第一继承者的情况发生，此时就需要进行冲突处理。进行冲突处理的主要依据的集群节点中存储的信息表的时间戳，由于信息同步机制在每次进行同步时都会带上当时集群Master节点的时间戳，所以，如果两个节点上的信息表如果时间戳相同，其信息表的内容也必然相同。由此可见，根据时间戳进行冲突处理是一个好的选择，它能够准确的判断出那个节点上的信息表是最新的，这就代表着该信息表是有效的。

* + 1. 心跳检测与信息同步模块

心跳检测机制是进行故障处理的前提，通过心跳检测机制，可以及时发现集群中有哪些节点失效，以便及时采取相应的处理措施。而信息同步则是为了保障集群元数据的安全性，将信息同步与心跳检测机制相结合是为了提高网络传输效率，同时简化系统设计。在论文第四章详细介绍了课题采用的结合同步信息的双向心跳检测机制，给出了其主要思想、模块的详细设计框图、采取的数据结构设计并对其性能进行分析与仿真。本小节主要对该机制的程序设计的整体流程做进一步的介绍和解释。

本模块的设计主要包含如下几个子模块：定时器、心跳信息的收发模块、信息表更新模块以及单点信息同步模块。首先给出本模块的整体流程设计，如图5-9所示。



图5-9 心跳检测&信息同步模块流程图

在图5-9中，心跳信息的发送模块作为定时器的回调函数，当计时器结束时，会调用信息发送模块进行数据整合和发送等操作。心跳信息的收发模块是直接采用开源软件Nanomsg进行底层通信的。Nanomsg是基于TCP/IP协议的消息队列，它提供了6中通信模式，包括一对一通信（Pair）模式、多对多（bus）模式、发布者—订阅者（pubsub）模式、调查者（survey）模式、管道（pipeline）模式以及请求—响应（reqrsp）模式。在本模块中采用了pubsub模式进行通信。

信息表更新模块主要负责维护集群的三张全局表，并将这三张全局表的变更信息写入缓冲器供信息同步模块使用，其主要流程如图5-10所示。



图5-10 信息表更新模块流程图

集群中其他节点在进行信息同步时，可能由于某种未知的原因导致同步失败，为了保证信息同步的完整和集群信息的一致性，需要引入单点信息同步机制，当通过心跳机制同步失败时，便由单点同步机制对相应的节点进行信息的同步。单点信息同步流程如图5-11所示。



图5-11 单点信息同步流程图

图5-11反映了进行单点信息同步的整体流程，Master节点在确认集群中某个节点信息同步失败后，会针对该节点发起单独的信息同步，Master节点首先向该节点发送一个同步信号，与该节点建立信息同步用的连接，该节点在接收到同步信号后启动接收功能并作出应答，Master节点在接收到应答信号后，向该节点发送同步信息，然后该节点根据接收到的同步信息更新信息表，并根据更新情况向Master节点回复ACK信号，ACK信号包含成功（success）和失败（fail）两种情况，如果同步成功，则结束此流程。

* + 1. 节点上线处理模块

Master节点的一个重要任务是建立、管理集群，并对集群信息元数据进行维护，进行这一切的基础是Master节点必须能对加入集群的机器进行判断并作出相应的处理，节点加入处理模块就是为了完成这个工作而设计的，它是集群组建的基础。本模块主要包含两个子模块：新节点上线处理模块和故障节点恢复上线处理模块。

在Master维护的集群信息表中含有集群节点的NodeID信息，在集群中，这个NodeID信息具有唯一性，且当集群节点发生故障时，并不会删除该节点信息，而是将其状态置位OFF状态，等待该节点恢复，因此可以根据上线节点的NodeID信息来判断某个节点是新节点还是由故障节点恢复而来的。图5-12为节点上线处理模块的整体流程。



图5-12 节点上线处理模块流程图

图5-12反映了对节点上线的处理流程，节点上线时，首先在集群内广播寻找Master节点，在得到Master节点回应后与其建立连接并发送包括节点相应的集群节点信息，如果是新节点，其集群节点信息，如NodeID、PI值等，均为NULL，Master节点在接收到节点信息后根据NodeID的值判断该节点是新增节点还是故障节点恢复之后上线，然后进入相应的处理模块。

如果上线的节点是故障节点恢复上线，则Master节点需要更新信息表并对其进行信息表的同步，并进行容灾恢复处理，主要是恢复节点上的数据，主要包括由于故障而丢失的数据以及在故障期间应当存储到该节点的数据，知道数据恢复完成后才认可该节点完成上线，可以正常运行参与集群任务；而如果上线的是新节点，Master节点需要为其分配集群节点信息，并进行信息同步，然后需要进行数据均衡处理，主要是根据数据负载均衡策略对当前集群存储的数据进行负载均衡，待数据均衡完成后，Master节点才认为该节点上线完成，可以正常运行和参与集群任务。

* + 1. 节点故障处理机制

集群是由一组自治的计算机组成的计算机集合，它们通过网络互连实现资源的共享和协同工作。通常情况下，为了控制成本，用于构成集群的计算机都是比较廉价的，其本身性能和机器质量并不是很好，容易由于种种原因而出现机器失效的情况。也就是说，在设计集群管理系统时，认为集群中的机器节点出现故障是必然发生的事情，无可避免。前文提到的心跳检测机制能有效地发现节点的失效，但是该模块只能进行告警，并没有相应的处理措施，而节点故障处理模块就是用来解决这个问题的。在现实场景下，机器发生故障的形式多种多样，比如网卡损坏、硬盘损坏、机器死机等，论文并不针对这种种的具体情况进行分析，而是从宏观的角度来看待节点的故障，即论文只针对损坏的节点的角色来分析具体的处理方案，而不会面面俱到的讲述每一种故障的发生及其应对措施。

在集群中，节点角色主要分为普通节点和Master节点，对节点的故障处理主要通过心跳检测和信息同步模块、选举模块配合外部的任务调度策略以及存储策略中的数据恢复算法来进行。当Master节点通过心跳检测模块检测到普通节点失效时，Master节点会调用信息更新模块，在集群信息表中更改其运行状态信息，即将该节点的运行状态从ON状态变更为OFF状态，如果当时该节点上有计算任务，还需要根据相应的任务调度策略进行任务的转移，当节点修复后重新上线时，需要根据节点的具体损坏形式对该节点的数据进行恢复，如图5-13所示。



图5-13 普通节点故障处理框图

当集群中的普通节点通过心跳检测模块发现Master节点失效时，会启用选举模块的为集群选举出新的Master节点来接管工作，并根据外部的任务调度策略选择中断集群中的外部任务或者进行集群中任务的转移和重组，在原来的Master节点修复后重新上线时，同样要根据其损坏情况进行数据的恢复，如图所示。



图5-14 Master节点故障处理框图

* 1. 功能测试与分析

系统的功能测试主要是你对集群管理系统的功能模块以及是否能正常运行并成功建立集群等方面进行测试。

* + 1. 测试目的

由于整个系统设计工程较大，难免会在处理模块上的细节上出现疏漏，因此需要对系统和相应的模块开展测试工作，争取在最大程度上发现集群管理系统存在的错误和缺陷，为后期工作的开展奠定一个良好的基础。

* + 1. 测试方法

在每个模块完成之后需要进行单元测试，主要目的是检测模块流程是否完整和流畅、模块功能是否正确。单元测试需要从程序的内部结构入手，通过测试结果寻找功能模块中可能出现的差错。

当功能模块编写并测试完成之后，需要对模块间进行联调和系统测试，以保证模块之间数据接口的准确性和系统功能的正确和完整性。在进行联调和系统测试时，需要对模块之间的数据传输进行追踪和观察检验，以便及时发现模块之间数据接口的差错并进行改正，直到最后完成相应的系统流程，保证流程的正确和流畅。

* + 1. 测试结果

在模块基本编写完成之后，先对功能模块进行单元测试，然后给出系统的联调测试结果。下面依次给出在将各个模块进行整合之后，针对各个模块的主要功能进行测试的结果。为了方便，本课题采用了3台服务器作为集群节点进行测试，继承节点为2个，为了描述方便，此处对集群信息做如下约定，集群中的节点角色Roles的值为1时代表该节点为Master节点，值为0时代表该节点为普通节点，值为1-k时代表该节点为继承优先级为k的节点；集群信息中的Status为节点状态，1表示接正常运行，0表示节点失效。

（1）选举模块

在选举模块中，分为让贤和继承两部分，其中让贤机制的测试采用集群初始部署时的情况给出测试结果，集群初始部署时，先指定一台机器为Master节点，由于测试需要，选择一台性能较低的机器，并先行启动该机器，待该机器完成启动后再启动其它两台机器，待这两台机器加入集群后，Master节点就会启动让贤机制，借此模拟集群建立过程并对贤机制进行测试。图5-15给出了只有一台机器时的集群信息，



图5-15 初始部署时集群信息（单节点）



图5-16 部署&让贤完成之后集群信息

从图5-15和图5-16可以看出，当执行完让贤机制后，有集群中PI值最高的节点02取代原来编号01的节点担任Master节点，而01节点则变更为第二优先级的继承节点，可见让贤机制的功能可以正确执行。

为了对选举模块中的继承机制进行功能测试，将当前Master节点与集群的连接断开，完成继承机制之后的集群信息表如图5-17所示。



图5-17 执行继承机制之后集群信息

从图5-16和图5-17可以看出，原来的Master节点失效后，执行继承机制，有原来的第一优先级继承者03节点取代02节点担任Master节点，且将02节点状态置位0，表示该节点失效，而01节点则晋升为第一优先级继承者，可见，继承机制可以正确执行，其功能完备。

（2）心跳检测与信息同步模块

此模块的主要功能的监控集群节点是否正常运行，并进行集群信息的同步，从图5-16和图5-17的对比中就可以看出该模块能有效监控集群节点运行状态，此处不再赘述，图5-18和图5-19分别给出了Master节点发送出去的心态信息和集群普通节点发送给Master节点的心跳信息。



图5-18 Master节点发送的心跳信息

图5-19 普通节点发送的心跳信息

在图5-18中，keepAlive为1时代表Master节点正常运行；flagInfoNode为1表示该信息表有变更，而flagIT和flagAT则表示其对于的信息表为发生改变；changeInfoNode后跟的内容是该表的变更内容，num等于2表有两个节点的信息发生改变，随后每一行代表对应节点的某些信息改变之后的内容，节点在进行信息表更新时，根据nodeID寻找到相应节点，并用其后的内容替代原有的相应内容即可。而在图5-19中，除了标识keepAlive信号外，还给出了相应节点的内存使用率和CPU使用率。

（3）节点上线处理模块

从图5-15和图5-16的对比可以看出，集群能正确检测到新节点上线，并将其加入到集群当中，而图5-20则给出了故障节点重新上线完成之后的集群信息。

图5-20 故障节点重新上线成功

从图5-17和图5-20中可以看到，编号为02的节点已经恢复并重新上线成功，由于是故障节点重新上线，所以并没有触发选举机制重新选举Master，但是02节点取代01节点成为第一优先级的继承者，而01节点则成为了次优先级的继承者。由此可见，该模块的功能正确。

* 1. 本章小结

本章首先介绍了系统的应用场景，并论述了系统采用的网络模型，给出系统的逻辑拓扑和物理拓扑；紧接着对系统进行功能划分，分析了系统的主要功能模块组成，在此基础上给出了系统的整体程序设计框图；然后详细介绍系统核心模块的具体设计；最后对系统功能进行测试，并给出测试结果，结果显示，系统能正确建立集群，并对集群各个节点的运行状态进行监控，当集群节点出现故障时，系统能及时进行处理，系统的核心功能没有出现严重错误，预期的功能也全都完整。

1. 结论与展望
   1. 全文总结

随着互联网技术的发展，集群技术被广泛应用于各种各样的场景之中，对集群性能及其管理和维护提出了更高的要求。论文以海量结构化数据的列式存储为背景，从集群管理系统的关键功能出发，深入研究相关算法，并对其关键模块进行设计与实现。在深入研究欺负选举算法及其改进算法的基础之上，在论文第三章提出了一种基于节点性能和低通信开销选举算法，在该算法中引入了让贤和继承的思想，第三章还给出了对该算法的性能分析与仿真，结果表明该算法能有效降低选举过程的通信开销；论文第四章中，在对传统的心态检测模型的研究基础上，结合具体需求，给出了基于PUSH&PULL模型的结合同步信息的双向心跳检测机制，并对该算法继续分析与仿真，结果表明，与PULL模型相比，该算法在通信开销和网络传输效率上均有所优化。

论文的主要工作内容包括：

（1）基于节点性能的低通信开销选举算法。论文深入研究了传统的欺负选举算法及其改进算法，针对其算法通信复杂度高、通信量波动较大等问题，提出了一种基于节点性能的低通信开销选举算法，该算法以节点性能表为核心，分为让贤和继承两大机制，能有效地降低系统的通信量

（2）结合同步信息的双向心跳检测机制。论文深入研究了传统的心跳检测模型，结合集群的具体特点和需求，设计了一种基于PULL&PUSH模型的结合同步信息的心跳检测机制，能有效检测节点故障，同时提高网络传输效率。

（3）集群管理系统及其关键模块的设计与实现。以上述的选举算法和心跳检测机制为核心，设计并实现了集群管理系统，并对系统功能和关键模块功能进行测试。

* 1. 展望

由于论文工作安排和时间有限等原因，本论文依然存在着一些问题和不足之处，有待在今后的研究中进一步改进和完善。

（1）论文侧重于对集群节点的监控、管理以及故障的处理机制的研究和设计，集群管理系统还有另外一个重要模块——任务调度，在本论文中并没有涉及，在后续工作中，可以结合相应的分布式存储策略，设计合适的任务调度策略，实现集群任务的本地化；

（2）目前，系统的管理对于管理人员的技术要求较高，在人性化方面不够完善，后续可以加入web前端系统，降低管理人员的操作难度，便于管理，同时也能更好的对集群进行监管；

（3）论文的系统设计只针对单机房，但是当前企业的集群部署大多分布在全国各地，而不会集中在某一个区域，对系统进行扩展，使其可以跨机房、跨网段进行集群部署是系统以后的改进方向之一。

参考文献

1. Gazal, Kaur P D. A survey on Big Data storage strategies[C]// International Conference on Green Computing and Internet of Things. IEEE, 2015:280-284.
2. 顾瑜. 云计算环境下数据保护关键技术研究[D]. 清华大学, 2014.
3. 李挥, 侯韩阳. 分布式存储编码与系统[M]. 科学出版社, 2016.
4. 赵学超. 数据中心集群监控系统设计与实现[D]. 中国地质大学(北京), 2012.
5. shvachko K, Kuang H, Radia s, et a1．The Hadoop distributed file system[c]//Proc of the Symp on Mass Storage Systems and Techn0109ies. Piscataway, NJ: IEEE, 2010: 1-10.
6. Zhuo C, Yang X. High available software architecture based on cluster technology[C]// TENCON '02. Proceedings. 2002 IEEE Region 10 Conference on Computers, Communications, Control and Power Engineering. IEEE, 2002:327-330 vol.1.
7. 刘思尧. 基于Linux平台的高可用集群管理系统的研究与实现[D]. 西北大学, 2012.
8. 顾梦非, 徐炜民. 一个基于Linux的集群部署方案[J]. 计算机应用与软件, 2008, 25(1):102-104.
9. 汪高翔. Linux高可用集群的研究与实现[D]. 北京邮电大学, 2015.
10. 陆嘉恒. 分布式系统及云计算概论[M]. 清华大学出版社, 2011.
11. 冉冉. 基于Linux环境的集群管理系统的设计与实现[D]. 电子科技大学, 2015.
12. 林伟伟, 刘波. 分布式计算、云计算与大数据[M]. 机械工业出版社, 2015.
13. 熊超. 分布式OLAP系统关键技术研究[D]. 桂林电子科技大学, 2012.
14. Garcia-Molina. Elections in a Distributed Computing System[J]. IEEE Transactions on Computers, 1982, C-31(1):48-59.
15. MohammadReza, Nasser, Mehdi EffatParvar, et al. Improved Algorithms for Leader Election in Distributed Systems[C]. ICCET, 2nd International Conference, 2010, V2: 6-10. 5.
16. Gholipour, Kordafshari, Jahanshahi, et al. A New Approach For Election Algorithm in Distributed Systems [C]. Communication Theory, Reliability, and Quality of Service CTRQ. Second International Conference, 2009: 70-74.
17. 李晓婷, 何博雄, 钟联炯. 分布式系统中欺负算法及其优化[J]. 西安工业学院学报, 2004, 24(3): 210-214.
18. 王立军.一种改进的选举算法在分布式OLAP系统中的应用研究[J]. 河北软件职 业技术学院学报, 2008, 10 (3): 52-55.
19. 吴宁, 马义忠. 基于欺负算法的优化改进[J]. 计算机工程, 2008, 24(19): 118-120.
20. Kordafshari M.S, Gholipour M, Mosakhani M, Haghighat A.T, et al. Modified bully election algorithm in distributed systems [J]. WSEAS Transactions on Information Science and Applications, 2005, 2(8):1189-1194.
21. Mishra B, Singh N, Singh R. Master-slave group based model for co-ordinator selection, an improvement of bully algorithm[C]// International Conference on Parallel, Distributed and Grid Computing. IEEE, 2015:457-460.
22. Arghavani A, Ahmadi E, Haghighat A T. Improved bully election algorithm in distributed systems[C]// International Conference on Information Technology and Multimedia. IEEE, 2012:1-6.
23. Effatparvar M R, Yazdani N, Effatparvar M, et al. Improved algorithms for leader election in distributed systems[C]// International Conference on Computer Engineering and Technology. IEEE Xplore, 2010:V2-6 - V2-10.
24. Soundarabai P B, Sahai R, Thriveni J, et al. Improved Bully Election Algorithm for Distributed Systems[J]. Eprint Arxiv, 2014.
25. Kabashi Q, Zeqiri A, Zabeli M. The Reduction of Number Messages in Election Bully Algorithm.system[J], 1: 2.
26. 张水平, 李有峰, 童样,等. 云数据中心心跳检测与故障评估[J]. 计算机工程与设计, 2014(10):3386-3391.
27. 马新宇. 一种高可用Linux集群管理系统的设计与实现[D]. 西安电子科技大学, 2013.
28. Yang T W, Wang K. Failure detection service with low mistake rates for SDN controllers[C]// Network Operations and Management Symposium. IEEE, 2016:1-6.
29. Xu C, Zhuang Y, Zhu W. A cluster heartbeat algorithm with ARMA-based prediction[C]// International Conference on Computer Science and Network Technology. IEEE, 2014:218-221.
30. Li F F, Yu X Z, Wu G. Design and Implementation of High Availability Distributed System Based on Multi-level Heartbeat Protocol[M]. IEEE Computer Society, 2009.
31. Peng G J, Chen G M, Liu Z H, et al. A novel self-regulatory failure detection algorithm for distributed storage system[C]// International Symposium on Instrumentation and Measurement, Sensor Network and Automation. IEEE, 2013:674-678.
32. Xie X, Zhao Y, Wang G. An Adaptive Failure Detection Method with Controllable QoS for Distributed Storage System[C]// International Conference on Multimedia Technology. IEEE, 2010:1-5.
33. Noor A S M, Sirajudin E A. A Design and Implementation of Cluster Heartbeat Network for Efficient Fault Detection[J]. Journal of Telecommunication, Electronic and Computer Engineering (JTEC), 2016, 8(4): 101-105.
34. Ghemawat s, Gobioff H, Leung s. The Google flle system[c]//Proc of ACM Symp on Operating systems Principles. New York: ACM, 2003: 29-43.

致谢

两年半的研究生生涯已经步入尾声了，在这两年半的学习生涯里经历的一切仍然历历在目，有过困惑和迷茫，也有过欢欣和灵光一闪的愉悦。这两年半你的点点滴滴都是我人生中宝贵的财富，在这期间，我不仅仅是学到专业知识，还有为人处世的道理，值得我一生珍惜。即将踏出学校，在此向我的导师、同学和家人表示由衷的感激。

首先，我要感谢我的导师别红霞教授。从我大四，尚未正式步入研究生生涯开始，别老师就开始指导我进行学习，在开始研究生生活之后，别老师从项目的推进给了我很多的指导，在论文的写作上更是耐心地帮我纠正和帮助，这两年多来，别老师不仅仅在学业上孜孜不倦的教诲、培养我，为我以后的发展打下基础，同时在生活上也教导我为人处事之道，让我快速成长。同时，在这里我也要感谢于泓老师在专业知识上对我的指导，帮助我顺利地推进学问论文工作。

其次，我要感谢实验室的张雪坤博士、巫庭耀博士以及李开放、邱宇等同学，他们在我两年半的学习阶段给我提供了很多帮助。特别感谢张雪坤师兄在论文撰写等方面的指导和帮助，以及巫庭耀博士在我寻找工作时为我分享他的经历，为我提供了巨大的帮助。

最后感谢我的家人，他们一直以来都字啊给予我最大的支持，不断地鼓励我，是我内心最坚实的后盾。

感谢所有支持和帮助我的人，我会努力奋斗，不断学习；不惧前行，不忘初衷。

攻读学位期间取得的研究成果

[1] 许巡枝,别红霞. 一种结合同步信息的双向心跳检测机制.北京：中国科技论文在线 [2017-12-04].http://www.paper.edu.cn/releasepaper

/Content/201712-31.