单位代码：10610

学　　号：xxxxxxxxxxxxxxx

四川大學

硕士研究生学位论文

（专业学位）

题目： **论文标题可能很长**

所以你可以把一部分放到副标题里

培养单位：计算机学院

作者姓名：XXX

指导教师：XXX　教授

学位类别：工程硕士

领域名称：计算机技术/具体领域

论文答辩时间：二〇二〇年二月

学位授予时间：二〇二〇年二月

Title of graduate thesis could be very very long but do not worry it will break the line automatically……

A dissertation submitted to Sichuan University
in partial fulfillment of the requirements
for the degree of

Master of Engineering
in Computer Technology

By
Kevin T. Lee
Supervisor: Prof. XXX

Computer Science, Sichuan University, Chengdu, China

February, 2020

声　明

本人声明所呈交的学位论文是本人在导师指导下（或联合培养导师组合作指导下）进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得四川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谢意。

本学位论文成果是本人在四川大学读书期间在导师指导下（或联合培养导师组合作指导下）取得的，论文成果归四川大学所有（或联合培养单位共有），特此声明。

学位论文作者签名：　　导师签名：

签字日期：　　年　月　日　　签字日期：　　年　月　日

摘　要

Raft 是一种用于管理被复制的日志的算法。该算法提供与 multi-Paxos 等价的结果，与 Paxos 一样高效，但是他的结构与 Paxos 不同；这使得 Raft 比 Paxos 更易于理解，为构建切实可行的系统提供了更加坚实的基础。为了增强可理解性，Raft 将共识的关键元素，比如 leader 选举、日志复制与安全性。此外，Raft 增强了一致性程度，并减少了需要考虑的状态数量。一项用户调查证明了对学生而言，Raft 比 Paxos 更加易于学习。Raft 还为集群成员关系的变更引入了一种使用“重叠多数”来保证安全性的新机制。

关键词：研究生学位论文；写作指南；参考模板

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Mae-

cenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Keywords: graduate thesis;writing guide;reference template

第 1 章　绪论

共识算法可以让一组机器像一个整体一样步调一致地工作，还有能力在一些成员出现故障时保持正常。正因如此，这组机器可以在构建可靠的大规模软件系统中扮演关键角色。在过去的几十年中，Paxos 在关于共识算法的讨论中占据统治地位，大多数共识机制的实现都基于 Paxos 或受其影响。Paxos 也成为了教授学生共识算法的首要手段。

不幸的是，尽管有很多人试图让它变得更加和蔼可亲，Paxos 仍然相当难以理解。而且，它的结构需要进行一些复杂的改动才能支撑实际的系统。因此，不论是系统开发者还是学生都在与 Paxos 苦苦“斗争”。

在我们与 Paxos“斗争”之后，我们打算找到一种新的共识算法，为系统构建与教学提供更好的基础。我们的方法有些与众不同，因为我们的主要目标是可理解性：我们能否为工程实现定义一种共识算法，并用一种比 Paxos 更易于学习的方式描述它？此外我们希望这个算法能促进系统构建者的直觉的发展，这对系统开发者而言至关重要。重要的不仅是算法能工作，还有理解它的工作原理。

这些工作的成果就是一个叫做 Raft 的共识算法。在 Raft 的设计中，我们应用了特定的技术来改善颗粒接送，包括分解（Raft 分离了 leader 选举、日志复制、和安全）和减少状态空间（相比于 Paxos，Raft 减少了不确定性以及服务器之间可以确保彼此的一致）。一项由来自两所大学的 43 名学生参与的用户调研显示，Raft 的理解难度显著地低于 Paxos：在学习两种算法后，其中的 33 名学生能更好地回答回答关于 Raft 的问题。

Raft 在很多方面与现存的共识算法十分相似（尤其是 Oki and Liskov 的 Viewstamped Replication），但是它有几个新颖的特性：

- 强 leader：Raft 使用了相比于其他共识算法更重要的 leader 角色。比如，日志条目只会从 leader 流向其他服务器。这简化了对日志复制的管理，使 Raft 更加容易被理解。
- Leader 选举：Raft 使用随机化的计时器来选举 leader。这只在其它共识算法都需要的心跳机制基础上增加了少量内容，却让解决冲突变得更加简单快速。
- 成员关系的变更：Raft 中用于改变集群中服务器集合的机制，使用了一种全新的“联合共识”的方法。在转移期间，两种配置的多数会存在重叠。这允许集群在更改配置期间能够正常运行。

第 2 章　复制状态机

共识算法通常产生于复制状态机的语境中。在这个方法中，在一系列服务器上的状态机计算出相同状态的相同拷贝；并且即使一些服务器发生故障也能正常运行。复制状态机用于解决多种分布式系统下的容错问题。例如，大规模系统拥有单独的集群 leader，比如 GFS、HDFS 与 RAMCloud，通常使用单独的复制状态机来管理 leader 选举与必须能从崩溃中幸存的存储配置信息。复制状态机的例子还包括 Chubby 和 Zookeeper。

复制状态机通常用复制日志来实现，如图 1 所示。每个服务器都存储包含一系列命令的日志，其状态机按顺序执行该命令。每个日志都包含相同顺序的相同命令，因此每个状态机处理相同的命令序列。由于状态机是确定性的，因此每个机器计算相同的状态和相同的输出序列。

保持复制的日志一致是共识算法的任务。服务器上的共识模块从客户端接收命令并将其添加到其日志中。它与其他服务器上的共识模块通信，以确保每个日志最终包含相同顺序的同一请求，即使某些服务器失败。一旦命令被正确复制，每个服务器的状态机按日志中的顺序处理它们，并且输出返回给客户端。因此，一组服务器看起来就像一个高度可靠的单体状态机。

实际系统采用的共识算法通常由以下几个特性：

- 保证安全性（对于非拜占庭将军问题的情形：即在网络延迟、分区、丢包、重复发送、网络重排序等情况下，永不返回错误的结果）
- 只要有大多数服务器还在运行并且可以与其他服务器、客户端互相通信，系统就是完全正常运行的。因此，由五个服务器构成的集群可以容忍其中任意两台服务器的故障。假定服务器因停止而故障，他们在一段时间后会从持久化存储中恢复状态并重新加入集群。
- 不依赖时间来保证日志的一致性，有故障的始终和极端的消息延迟在最坏情况下可能导致可用性的问题。
- 一般情况下，一条命令可以在集群的大多数节点完成一轮 RPC 响应后完成。少数服务器的运行缓慢一般不会影响整个系统的性能。

第 3 章　Paxos 的问题

在过去的十年里，Leslie Lamport 的 Paxos 几乎成为了共识的同义词。它是最常在课堂中被教授的协议，大多数共识的实现也以它为出发点。Paxos 首先定义了一种能在单一决议上达成一致的协议，比如复制一个单独的日志条目。我们把这个子集称为 Single-decree Paxos。Paxos 随后将多个这种协议的实例联合在一起，以促进一系列的决议，比如日志（multi-Paxos）。Paxos 同时保证了安全性与活跃性，同时支持集群成员的变更。它的正确性已经被证明，并且在通常情况下十分高效。

不幸的是，Paxos 有两个重大缺陷。第一个缺陷是 Paxos 非常难以理解。它的完整阐述臭名昭著地模糊，只有很少的人在付出巨大努力后可以理解。因此，也有一些用简单的方式解释 Paxos 的尝试。即使这些解释专注于 single-decree 的子集，也同样面临很多挑战。在一个对 NSDI 2012 出席者的非正式调查中，我们发现很少有人多 Paxos 感到舒适，即使是在富有经验的研究者人群之中。我们也曾挣扎着研究 Paxos，在花费了近一年时间，阅读了好几种简化的解释并设计了我们自己自己的协议之后，我们仍然没有完全理解这个协议。我们假设 Paxos 的模糊性来源于他选择以 single-decree 的子集作为基础。Single-decree Paxos 复杂而精密，他被分成两个部分却没有简单直观的解释，而且两部分不能被独立地理解。正因如此，很难建立起对 single-decree 协议的直观认识。Multi-Paxos 的组成添加了重要的复杂性和精妙之处。我们相信对多项决议达成共识的问题（比如一个日志而不是单独的条目）可以用其他更加直接与明显的方式拆分。

Paxos 的第二个问题是没有为构建实用系统提供良好的基础。其中一个原因是对于 multi-Paxos 而言，没有一种公认的算法。Lamport 的描述大多数是关于的 single-decree Paxos 的。它给出了 multi-Paxos 的大概方法，但是缺失了很多细节。有很多充实并优化 Paxos 的尝试，但是它们与 Lamport 的描述、他们彼此之间都有区别。一些系统，比如 Chubby 已经实现了雷系 Paxos 的系统，但是大多数细节都还没有公开。

此外，Paxos 的结构不适合构建真实的系统；这是以“单个决议”分解问题的另一个后果。选择一组日志的条目，再把他们合并为一个连续的日志没有任何好处，这只能增加复杂性。围绕日志构建一个新条目只能在某种约束下按顺序扩展的系统会更加简单和高效。另一个问题是 Paxos 在他的核心中使用一种对称的点对点方式（尽管这最终被认为是为优化性能而选用的弱 leader 形式）。这在只需要做出一个决议的简单情况下是有道理的，但是很少有真实的系统会采用这种方法。如果有一系列决议必须通过，更简单快速的方法是先选举出一个 leader，然后让 leader 来协调这些决议。

因此，实用的系统往往只有很少的部分相似。每种从 Paxos 出发的实现都发现了实现它的困难之处，之后开发出了相当不同的体系。这耗时且易错，而 Paxos 的理解难度加剧了这一现象。Paxos 的公式也许对有利于在理论上证明它的正确性，但是真实的实现与 Paxos 如此的不同，使得这些证明没有什么价值。下列的来自 Chubby 实现者的评

论就十分典型：

> Paxos 算法的描述与真实世界的系统有很大的差距.... 最终的系统将会基于一种未证明的的协议

由于这些问题，我们推断出 Paxos 没有为系统构建与教学提供良好的基础。鉴于共识在构建大规模系统的重要性，我们决定尝试自主设计一种比 Paxos 更好的共识算法。Raft 就是这一尝试的产物。

第 4 章　为可理解性设计

参考文献