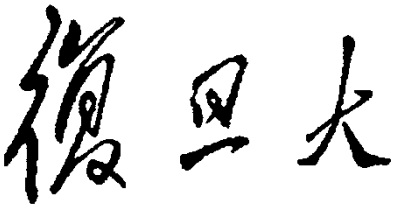


本科毕业论文

论文题目：信息中心网络缓存算法性能分析

院　　系：信息学院电子工程系

专　　业：电子信息科学与技术

姓　　名：许灿　　学　　号：12307130160

指导教师：徐跃东　　职　　称：副研究员

2016 年 05 月 22 日

摘　要

当代互联网的应用方式正从面向主机进行点对点通信转向终端用户对海量内容资源的获取。为了适应此转变，研究界提出了多种以信息与内容为中心的新型网络架构。这类网络架构中最重要的特征是利用网络内置的缓存系统提高接收者索取内容的传输效率和网络基础资源的利用率。与传统的分层互联网缓存、内容分发缓存相比，信息中心网络缓存系统呈现了缓存透明化、普遍化和细粒度化等新特征，这对缓存系统的建模、行为理解和优化方法都提出了挑战。在介绍了内容中心网络的新特征及带来的挑战后，本文探讨了不同替换策略与决策策略在缓存系统中的应用，分析了缓存网络的常见优化方法，验证了一些学界之前的既定结论，也对不少现有理论设想提出了挑战。本文最后对信息中心网络缓存系统仍待解决的关键问题作了分析并预测了未来的发展。

关键词：信息中心网络；内容中心网络；缓存；替换策略；决策策略

Abstract

The modern Internet application is focusing on the communication and transmission of massive content resources to the terminal users instead of host endian-to-endian communication. In order to adapt to this change, the research community has put forward a variety of new network architecture based on information and content called ICN. The most important feature of this kind of network architecture is to improve the transmission efficiency and the utilization of network infrastructure resources by using the built-in caching system. Compared to traditional hierarchical Internet or content distribution network cache system, information center network caching system presents the transparent, universal and fine-grained features. These features challenge traditional cache system model, behavior understanding and optimization method. After introducing new features and challenges of the content centric network, this paper discusses some different replacement and placement strategies in ICN cache system with analysis of the common optimization method of Web caching. Some scholars' underlying views are verified, while many previous theories are challenged. Finally, key issues of ICN cache system to be resolved are analyzed while forecasting the future trend.

Key words: information-centric network, content-centric network, cache replacement algorithm, cache placement algorithm

目录

依据思科 Cisco（全球最知名的网络解决方案提供商）的统计数据,全球 IP 流量(Internet Protocol，泛指 TCP/IP 网络架构中的总数据传输流量)在过去的五年中翻了两番。2012 至 17 年这五年时间里,IP 总流量仍以 20%多的平均年化复合增长率实现快速增长[1]。而且，绝大部分的网络流量都由内容获取类型的应用产生，据估计到 2016 年，视频类应用总流量将占所有网络流量的 86%。未来，随着社交网络普及、网络带宽提升与单位流量费用的下降，用户生成内容（UGC）如社交分享性自拍、网络电视、高清视频点播(VOD)需求增长，用户驱动的交通生成的数字视频内容将继续在未来几年快速增长[2]。

参考 W3C 的统计，大多数的互联网应用从发送方主动 PUSH 驱动用户方接受端对端的通讯模式转向接受者（即互联网用户）主动请求以查询海量内容资源。面对这样的改变与发展趋势，我们应当在网络架构的整体体系层面上提供易扩展和高效率的获取内容原生性支持并方便移动端、保障安全性。学界近年依赖提出一种以内容/信息作为中心的新型网络架构体系[3,4, 5, 6, 7, 8, 9]，即 information-centric networking，简称为 ICN。具体地有 DONA[4],ICN[5],与 PSIRP[9]等，这些在参考文献内有详细论述。考虑到我们实现 ICN 的基础架构并不依赖这几种具体设计，因此本文不多讨论，有兴趣的读者可以专门研究[10, 11, 12]。

在 ICN 体系架构的国际网络中，实际的使用者即用户并不需要关心 IP 地址的位置或者说内容资源的位置与具体文件名，只需要了解内容本身到底是什么。一个统一身份、定位，基于内容路由和传输将提高内容的地位，使其成为 ICN 中的核心资源、网络中的“最高标识”。此种网络系统架构提供通用的、透明的、无处不在的网络内置高速 CACHE，来加快内容的传输并增加网络系统资源的利用效率[10]，最终可缓解网络带宽使用量的急速增长对网络资源造成的严重压力。然而，由于相对封闭的缓存系统和缺乏统一标识系统的缓存机制等原因，系统潜在的缓存能力并不能充分实现——例如，作为统一网络对象标识符的 URL，也作为一个定位符来使用。当相同的 WEB 对象被放置在不同内容提供商 ISP 的服务器中时，将使用不同的网址 URL 以确定它的位置并访问相应服务器（集群）。而这却导致网络缓存系统将识别这两 URL 对应内容为相异的网络资源对象。

众所周知，网络与内存中的缓存理论以及策略优化、具体实现已经被反复研究过，并且在 Content Delivery Network（即内容分发网络 CDN）与 Peer-to-Peer(即点对点 P2P)以及大众化的分布式 Web 体系中实现了应用,最终提高了网络体系的效率。ICN 中部分概念、术语与之前的网络缓存理论类似，但是与 CDN 相比却也有非常多的不同，尤其是在具体实现方面。这是由 ICN 缓存系统的特性所决定的，究其原因主要是内容核心性、普遍性、透明性等重要性质反映了当代网络流量的急迫需求与之前理论/模型的不足之处。本文专注于 ICN 缓存的研究，对

ICN 缓存目前理论、策略加以介绍分析，对具体的研究方法、思路进行比较和批判，以此明确将来的研究方向。更进一步地，我希望能将缓存策略应用到 ICN 块粒度的仿真与分析上，通过利用按照 ICN 规则生成的海量块级别请求数据进行仿真，本文比较了诸多算法、设计方法的区别与性能优劣，实现对内容资源粒度块的有效利用，增加了命中率与网络基础资源使用效率。

因此，本文逐渐展开：第一章将对 ICN 缓存研究进行综述,着力于 ICN 架构中缓存系统所呈现的新特点与新问题。第二章主要介绍 ICN 网络系统的缓存替换策略并进行标准化测试以对比其性能。第三章，主要介绍 ICN 网络系统的缓存决策策略并进行性能测试。最后，第四章对全文进行总结，对 ICN 网络的研究发展作出一定的分析预测。

第 1 章　ICN 缓存研究概述

ICN 网络架构里,缓存显现出有别于传统的 Web 万维网,点对点 (Peer-to-Peer，简称 P2P), Content Delivery Network(内容分发网络，简称 CDN) 等封闭的缓存系统的新特性,对原先的缓存理论中常见算法和优化策略均进行了挑战。总而言之，ICN 中缓存网络的新特性有三点：缓存普遍化（ubiquity）、透明化（transparency）和细粒度性（Fine-granularity）。以下的内容会具体解释这三个新特性，并指出其对传统 Cache 理论算法与优化策略之挑战，点明 ICN 缓存系统的重点研究对象与研究方法。

1.1 ICN 缓存的新特性及研究挑战

1.1.1 缓存普遍化（Cache ubiquity）

考虑到传统的网络缓存系统中对象/数据存储后位置固定，并且通常采用树的层次结构来存储对象内容，缓存位置之间的协同运作和资源放置策略建立在相应数学模型的基础上。而模型则事先考虑现有的业务需求以及具体数据的内容，使得之后的数据传输缓存方案也接近于最佳的解决方案。知识和高速缓存结构，并获得最佳解决方案。可是由于 ICN 以内容为核心，同一资源可以存储在不同服务器集群以及不同的服务提供商，缓存变得普遍化、透明化，具体缓存点通常也非固定，缓存系统具体的拓扑网状结构可用描述比较通用的图来描绘，网络节点之间的上游和下游的 HOP（网络流量跳跃）关系不明显。所有的这些特性都提高了数学建模和预先分析设计的难度，也使得高速缓存间的显性协作变得越来越困难。无处不在的缓存也需要内容资源更加具有可用性(availability)。

在以前的万维网 Web Cache 或 CDN Cache 中，请求某缓存对象后是否可用很明显。对 CDN 网络来说，可以根据历史数据预测将来接入的具体需求，并在网络结构中对请求对象资源进行提前 Push 和 Cache 部署以应对将来的请求流量。在 CDN 中还可以通过 Redirect 重定向与 DNS 解析等方式来确保资源副本全局可用。在 Web 分层结构中，一次指定的资源请求在层次结构的网络缓存系统中，只有在缓存的请求点到网络缓存根节点 Root 上的路径可用，在该路径外部节点的资源对象对该请求均不可用。可是在 ICN 体系结构中，缓存之普遍性和内容资源对象被缓存驻留时间之短暂使得 ICN 网络系统具有高度的动态性。传统的全局解析/路由系统进行统一调度并向所有 Node(节点)Post（推送）所有网络内置缓存内容节点缓存池中的资源对象可用性，但这使得每次插入后更新消息之规模

很大，以至于系统的可扩展性得不到保障。并且，ICN 系统高度的动态性让维持网络状态的总体一致变得特别困难。目前 ICN 架构缓存网络急需解决的问题就是——怎样在动态性强的普遍网络缓存内置环境下保持一定的对象可用性并且减少对象资源的获取成本。这些都是 ICN 网络架构缓存系统急需解决的现实问题。

1.1.2 Cache transparency(透明化)

传统的网络 Cache 系统针对某一种固定流量类型来设计，一般是专用的封闭网络系统，如 Web, CDN（内容分发网络），Peer-to-peer(P2P)等缓存系统。最常见的 Web 网络系统缓存是建立在公开的 HTTP/SSL 协议之上。由于 Web 内容资源基于域（Domain）由各网站自主命名，同一个内容对象无法被识别为统一的内容资源，这导致内容对象实际上按域进行隔离命名。P2P 网络应用一般基于私有的协议/协议栈，这让不同的 P2P 应用采取不同方案且各自独立封闭，无法实现相互间 Cache 空间的共享，资源无法有效利用。最近的研究人员也在寻找解决方案以应对这种缺陷，P2P 设计上也有缓存透明化的趋势，比如这几年 DECADE Workshop 进行的研究工作[13, 14]。DECADE 也希望能建立共享的网络缓存层面的基础设施，使得任何应用都可以独立地管理所需的网络缓存资源与空间。然而，P2P 天生缺乏的命名一致性使得 DECADE 并没有找到不同的应用间缓存对象的共享方法。DECADE 研究中没能解决的上述难题应该归结于两大原因：其一是没有一套统一命名内容对象的方法；其二是全网络基础设施间采用的通信协议不全是公开一致的，难以在底层协调，只能在应用层面各自找解决方案。

这几种传统网络架构都不能很好地应对现代互联网发展对网络基础设施提出的挑战。因此 ICN 为此而设计并妥善地解决了前面所述之问题：第一，ICN 网络对内容资源实现了全局的统一标识，这保障了命名的总体一致性。而且，命名本身是自洽的，含有足够的信息以自我验证完整性、正确性[4, 7, 8, 9, 10, 15]，使得内容对象更加安全，利于网络中个体的安全保护以及整体的健壮性检测；第二，ICN 通过统一的内容标识符来判断对象身份，并在网络中进行对象路由与存储决策。在网络层而非应用层识别内容对象身份标识的方法实现了应用与资源标识的解耦，完全使缓存决策不受具体应用的影响，让路由与缓存成为普适的、公开透明的、中间件对上层高可用的网络基本服务。

ICN 缓存的透明化同时也对设计、实现提出了更多要求，我们也需要让系统具有如下性质：

1.1.2.1 ICN 缓存内容对象应用间的有效竞争性共享

不同的应用所请求的的内容资源对象很不同，通常高度异质化，比如说常见的 Web 对象、Vod 视频对象、用户自产的内容资源对象(UGC)与人际之间传输共享的文件对象。这些对象在数量的规模、对象大小、对象流行度方面有着天壤之别[17] 。举例来说：Web 对象数量很多，数量的规模很大，在十亿以上，但每个资源的大小都比较小，属于小质多量类型；VoD 视频对象数量不算多，统计上大概是几十万上百万的数量级别(考虑到人类拍影视作品的出品速度并没有指数级别增长，全球网络中 VoD 视频的数量级并不会很快增长，而且越增会越慢)，然而每个视频 VOD 对象都很大，两小时的电影 1080P 级别的清晰度就有几十 GB。内容资源类型不同导致的缓存系统设计上得去兼顾各种需求，对传统封闭式固定设计方案的缓存系统进行了一定程度上的淘汰，也对新一代 ICN 异构处理能力提出了很高的要求。我们设计的 ICN 网络架构应该可以在异构的应用间高效共享网络基本缓存资源与设施，使得新一代的 ICN 更加易于应用的开发与应对流量增长。

1.1.2.2 网络系统缓存不同目标之间的兼顾

目前主流的三种缓存系统的对象与实现目标较为单一。在 Web 万维网架构中缓存主要是应对高频的 Web 网络流量，而这依赖于降低网络传输内部流量和最终用户端的实际延迟；P2P(Peer to Peer)的缓存是应对单源大数据流的场景，实际上可以视为大储量文件的有效分发，使用目的是降低源节点服务器的负载与所需网络带宽与流量 [16] 。ICN 是下一代网络的基础架构与服务设施，应该能够处理多种网络流量类型，涵盖 Web、大文件传输、VoD 视频等不同应用类型，这些场景中应用都将直接运行于 ICN 之上。异构的缓存系统考虑的目标不完全相同，ICN 可以兼顾不同缓存目标，但是要有核心标准与规范，合理地评估整体系统的效率与功能。

1.1.2.3 ICN 缓存系统能以线速处理新对象

在 ICN 网络内部，内容对象缓存的透明化对具体节点中缓存执行速度构成了挑战。这要求 ICN 网络中的缓存系统以线速来执行，这与一般意义上利用硬盘来存储暂时数据的缓存系统相比，差别很大[18]，是 ICN 缓存系统管理策略上的挑战。利用硬盘来进行持久化数据存储的缓存系统通常使用大型 Database(数据库)的“软件调度管理”模式，缓存策略工作在应用层，对于 I/O 没有速度上的很大要求，反而对同时并发处理量 TPS/QPS 要求较大。在实际使用中存储容量的

上限由软件系统自身策略、运维和管理人员优化能力决定，实际的存储管理算法以 B-TREE 作索引为主，读写处理与锁机制较为复杂。然而 ICN 中的缓存节点是网络层中互相串联的基础设备，必需要利用内存缓存数据加速读写，以线速处理新请求，处理速度的上限在于内存的访问读写速度，这使得节点中缓存调度策略简单、直接、有效[19]。

1.1.3 缓存细粒度化（Fine-granularity)

目前的缓存系统一般以文件或文件的拆分小段（如视频网站分段缓存）为最小单元进行内容资源对象的存储、读取和替换等原子操作[16, 20, 21]。可是 ICN 需要以线速处理单元对象[22]，以文件作为原子单元来进行缓存 I/O 读写的时间开销非常大，达不到线速执行的要求。同时缓存中的内存容量可能无法装下文件，这也不满足通用化需求。因此，ICN 系统一般将文件标准化分割为独立的可标识数据小块(chunk)，将数据块视为缓存基本原子单元[5,22, 24]。缓存粒度上的标准化产生了新的变化与需求：

1.1.3.1 流行度的变化难以拓展

针对文件粒度 Web 请求的流行度已经有较为成熟的研究。比如：Web 中的文件对象的流行度服从 Zipf 分布，P2P 中视频对象的流行度按照 mandelbrot-zipf 分布[21]。但文件粒度的访问流行度不能直接拓展到块粒度流行度的分布。因为同一个文件的不同块被访问的频率不大相同。比如说，手机用户在观看视频或者直播时，一般会观看开始的一段时间（对应前面的块），并决定是否继续观看下去。通常情况视频前面的块热门，而直播一段时间后黄金时段的块较热门，这导致同一个视频文件中不同块是有不同的访问流行度，无法直接现有将结论推至块级别。遗憾的是，目前块级别访问的流行度分布特性还没有可信的理论模型与详细的实证研究可以采纳。

1.1.3.2 请求文件独立参考模型无效，请求具有关联性

之前以文件级别的缓存模型一般基于请求文件独立参考模型(independent reference model)这一假设，即某个对象（一般指文件）被请求的概率只与该对象的流行度有关，而之前的状态（即请求序列）无关联。然而，在块层面同一文件的不同块之间存在顺序性、关联性，用户对块的请求通常以线性顺序进行。请求独立参考模型基本假设已经失效，这也使得许多已有缓存理论的分析方法与结论不能适用于块粒度 ICN 缓存系统分析。

1.1.3.3 缓存空间的利用应更加有效

将大的文件划分为标准化块单元后，同一个文件不同块可以从不同网络缓存节点来获取，这极大地增加了带宽的利用率与块获取路径的调度选择余量。在缓存命中以及替换策略上去替换标准化的块的效率也远比替换大文件（块）的效率高，提升 ICN 网络内部调整的速度。不过，以块为原子单位进行缓存，也让一些基于文件大小来决策的缓存替换算法无法适用。

1.2 ICN 缓存研究方向与具体内容

ICN 缓存研究方向有两条：第一种是网络缓存系统具体性能的优化策略与方法，此种研究关注于可实现的具体技术解决方案，从不同的层面与具体特性着手优化 ICN 缓存系统的整体性能；第二种是对 ICN 缓存网络的基本规则适度抽象和简化，提出对应理论模型，并进行性质分析与定量推演，作为理论基础便于理解 ICN 缓存系统中具体现象与行为。

下图列下了 ICN Cache 系统中大体研究方向与具体研究内容：

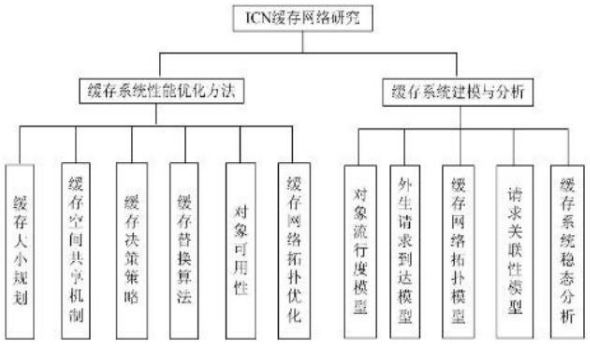


Fig.1 Research topics of ICN cache network

图 1 ICN 缓存网络研究内容

1.2.1 ICN 缓存系统性能优化方法

由于缓存的内置化、普遍化是实现 ICN 网络架构的主要手段，因此 ICN 网络应该就具有内在易扩展性，优化 ICN 网络缓存的性能也愈发重要。优化 ICN 缓存性能的方法有：网络缓存节点存储容量的合理设置、缓存节点中内容资源的跨应用共享、请求对象缓存节点的确认（即 Placement）、缓存不命中时的替换（即 Replacement）算法、缓存资源对请求的可用性决策与 ICN 网络的拓扑优化。目前学界除了 ICN 网络的拓扑优化没有可信结论与成熟方案外，其余四个方面已经有不少研究。

1.2.2 ICN 缓存系统建模与分析

ICN 网络缓存建模目前难点是多种新的特性要在不确定网络拓扑结构的情况下同时满足，而且网络缓存节点间有着复杂的联动关系。

总体来看，ICN 网络缓存目前的建模水平与可信度都不高，研究这方面有较多成果的学者也不多，应该是以后的发展方向。

第 2 章 ICN 中缓存替换策略

本章主要研究了在 ICN 网络中，缓存在节点中的替换策略对网络性能的影响。虽然目前网络上的信息流量超过七成为流媒体（视频为主）且此比例仍在不断上升产生了大量的真实数据，但是 ICN 网络还没有真正投入运行，块级别缓存的应用也没有出现。故本文主要靠模拟网络访问数据来衡量不同的缓存替换策略以及缓存决策策略在 ICN 网络中的性能，最终得出了一些新的结果。未来的互联网基础架构必然会有变革，缓存在互联网中的作用只会更加重要，因此学界对 ICN 缓存的研究是非常有意义的。

2.1 ICN 缓存替换策略介绍

缓存替换策略如 LRU、LFU 等比较知名，在 ICN 网络的仿真中也应该被测试其性能以正确评估。然而，由于 ICN 的一些特性，部分缓存策略不能适用，自然也没有测试的必要，我们应该先予以排除。

比如说：ICN 中 chunk 级别大小标准化，SIZE 大小一致，因此原先很多基于缓存对象大小决定是否替换的策略不再适用。SIZE 与 LRU-Threshold, Log(size) LRU, Lowest-Latency-First 不需要再次被研究，因为具体语境已经发生了变化。在之前的研究中表明，真实 WEB 缓存架构里大多数情况下，LRU 都要比 LFU 好。只有在极少情况下，LFU 表现比 LRU 好。我们需要针对 ICN 网络的特定环境重新审视此结论。

而 MRU 策略这种越久远内容越可能最近被用到的情况显然不符合内容关联度较大的 ICN 网络，因此也不如其他策略，不需要测试。

综上所述，我选取了最常见且能在 ICN 网络中被利用的六种替换策略：LRU/LFU/RANDOM/CLOCK/LIRS/ARC 来对比其性能，并考虑最重要的两个指标：缓存的各级命中率以及平均取到需要数据的路径长度。

同时我们还要考虑缓存池的大小与单个缓存对象的大小。由于 ICN 线速通

常很大，比如 120 Gb/s，若每个缓存节点对象平均储存时间为 100ms,那缓存所需空间至少为 12Gbit = 1.5GB,SRAM 无法处理这么大的缓存池，因此 RLDRAM 是比较理想的选择。

在实际 ICN 网络缓存中，单个节点最大块对象存储数量可以在 800-1500 之间，但是目前比较替换算法性能，重要的是之前的性能指标，因此我们设置块的总数量是 1000，单个节点缓存池大小 10，网络上 30 个节点。

本章算法的具体实现可以见附录 ICN_XC/Cache_replace 文件夹下的各个 Python 模块，路径决策策略采用最常见的 LCE。

2.2 替换算法介绍与实现

2.1.1 LFU 缓存替换策略

LFU 是最近最不常用页面置换算法(Least Frequently Used),缓存池满了后先淘汰一定时期内被访问次数最少的块内容对象。

除了 init,repr 等内置方法的重新定义，我们只需要写查询与插入两个方法。

我的实现方法是利用小根堆来维护最少被访问的块内容对象，小根堆针对此种情景应该是算法复杂度最低的，平均每次插入替换仅需 O(log N),而线性有序的数据结构如链表/数组则需要 O(N)，堆在效率上最高。而读取查询只要在 LFU 对象中加入一个字典 dict(Python 内置数据结构，哈希表实现的字典)，维护对应 key-value 对即可。哈希表需要额外的内存来减少放置值的重复率并节省查询时间，但是我们的 ICN 单节点缓存上限也才 1000 左右，内存空间完全足以应对维护此字典对。

首先我要测试 LFU 模块的代码逻辑正确性：

使用 Rand 访问函数生成了 200 组的数据（在 Data_history 下的 LCE_LFU_Small 后缀的组里面），不同的块有 12 个，以方便手动验证，如下图 2-1 与图 2-2，其中拓扑图片采用 python 中 matplotlib 包产生

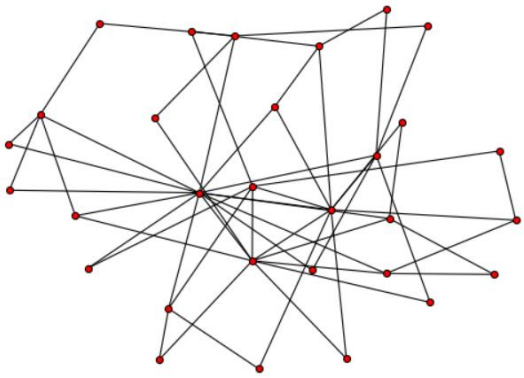


图 2-1 30 个节点的拓扑

```
['34.80.57.86', '223.101.153.74', '2']
L = Source user,R = Dest server,Find the reqFile
req all import successfully!
Hit times Total   = 27
ReqTimesTotal   = 200
Hit Ratio= 0.135
Hit times in the level 0 is : 14  ratio: 0.07
Hit times in the level 1 is : 10  ratio: 0.05
Hit times in the level 2 is : 1  ratio: 0.005
Hit times in the level 3 is : 2  ratio: 0.01
The average length of get: 3.2
Stat End Successfully!
Press any key to continue . . .
```

图 2-2 200 组小数据测试正确性

[0, 0, 16, 5, 4, 0, 1, 0, 0, 1, 2, 0, 0, 0, 1]，此列表即为拓扑中的度数列表，其中度数为 2 的有 16 个节点

其中 Topology 采用 BA 无标度网络模型，参数 N=30(即 30 个节点)，M=2（即平均每个节点度为 2），TOPO 中的 replacementSize=1（即仅缓存一个节点），规模比较小以方便验证。在图 2-1-1 中的 hit 指未到最终请求节点就在路径上找到了缓存，hit time in level 0 指在用户接入的第一个内容路由节点就找到了本地的缓存，总共有 14 组，平均得到需要内容对象数据的实际长度为 3.2（无论是否有缓存）。

此组数据缓存池仅一个对象，是为了方便跟之后的算法进行交叉验证。

值得关注的，平均路径长度为 3.2，虽然 M=2 让平均度数为 4，但是本图依然并非稠密，平均路径长度-1 后 2.2 才是第一个请求路径上的节点到取到数据点的距离。这在小规模 BA 网络中依然很小，主要归咎于为了便于跟之后算法进行对拍设置得缓存池大小 1 使得缓存的次数很少。

之后是设置请求文件的规律，由于随机性均匀分布的 chunk 块级别流行度不符合 ICN 环境，如部分热门块特别是热门视频的前面部分请求频率会非常高。虽然文件级别的 zipf 分布目前不一定能推至块层面，但是前面那些热门块占用了非常多的网络流量是肯定的。我们依然可以用 zipf 分布来模拟请求文件的数

量，但是 zipf 分布应该在基本符合 8-2 定理。

由 Zipf 分布的公式[22]得

$$p(x) = \frac{x^{-a}}{\zeta(a)},$$

其中ζ是黎曼函数（Riemann Zeta function），重要的是 numpy.random.zipf 直接给出了按此函数生成的列表，调用案例是 numpy.random.zipf(x,a)，其中 x 为列表长度（即所需要生成的列表有多少个数），a 直接决定曲线的平滑程度（即前面 20%的块被调用次数）。定义中 a>1

当 a>3 时几乎完全平滑，与均匀化随机差别不大，不符合预期。

当 a<1.5 时，最大的频率已经超过 0.85，也不符合预期。

我们发现 a=1.90 时比较符合，随机测试了多组，统计上期望接近 80%，但是标准差比较大，下面图 2-3 是其中一组结果。

```
[0, 0, 18, 8, 3, 5, 2, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1]
4506 5609
The 20% top sum / total ratio:  0.803351756106
Press any key to continue . . .
```

图 2-3 a=1.90 时的分布

此组数据内容块个数为 1000，详见文件 zipf_list_0.8_1000。然而虽然 a=1.90 满足了 2-8 原则，但是大多数块的频数仅 1-6 次，与前 20%相比过少，实际测试请求次数不过百万时出现次数可能少到难以被缓存。而且 2-8 原则在块级别上的存在还无可信理论。为了方便性能测试，我们可以采取 2-6 原则，即前 20%共计满足 60%热度。

测试后发现 a=2.3 较符合要求，多组时波动比较明显，但是加入统计函数后可以不采纳偏差大的实验组。下图 2-4 是其中一组结果

```
[0, 0, 16, 12, 3, 2, 2, 1, 1, 0, 1, 0, 1, 1]
a and N = 2.3 1000
1522 2443
The 20% top sum / total ratio:  0.623004502661
Press any key to continue . . .
```

图 2-4 a=2.3 时的分布

由于 a=2.3 时的数据不稳定，我们 zipf 分布在 x 固定时只需取一组合意的即可，因此可以事先 zipf 分布打印多组我们挑选一组，以实现结果的稳定与准确。在此我们依靠读写分离，实现了解耦，优化了系统架构，块内容具体分布度见文件。

在取定 a=2.3 后，我们应该拿大数据进行性能压力测试，我们设定文件块的

总数量是 1000，单个节点缓存池大小 12，网络上 150 个节点，M=4 即平均 8 度，这样几乎每个块都可以被缓存，请求总条数是 20000。结果如下图 2-5

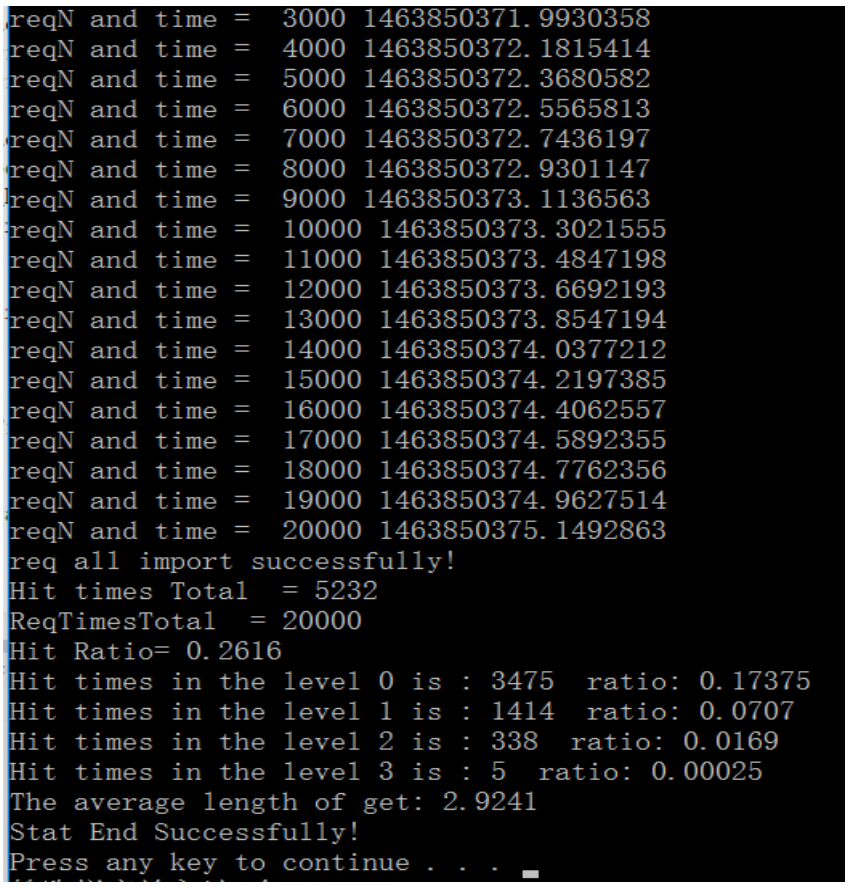


图 2-5 压力测试 20000 组时

这组压力测试后我发现命中率挺高，似乎比真实 ICN 网络中的高不少，主要原因如下：

一，BA 无标度网络，N=150 时取 M=4 过大，可以改为 3

二，2-6 原则依然让部分块过热，由于真实世界路由器数量以十万计而非几百计，应该将其降低至 2-5 原则且增加块数量至 3000

测试后发现 a=2.6 符合，如下图所示：

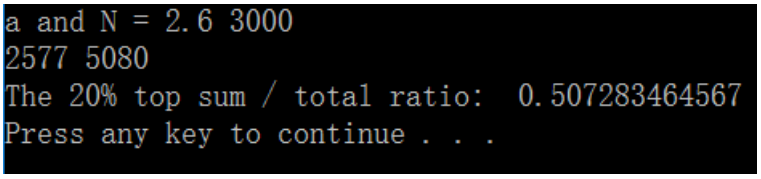


图 2-6 二五原则

最终的性能测试中，我们设定内容块的总数量是 3000，单个节点缓存池大小 12，网络上 150 个节点，M=3 即平均 6 度，请求符合 2-5 规则，很多块都可以被缓存，请求总条数是 500000。拓扑如图 2.7，最终结果如图 2.8.

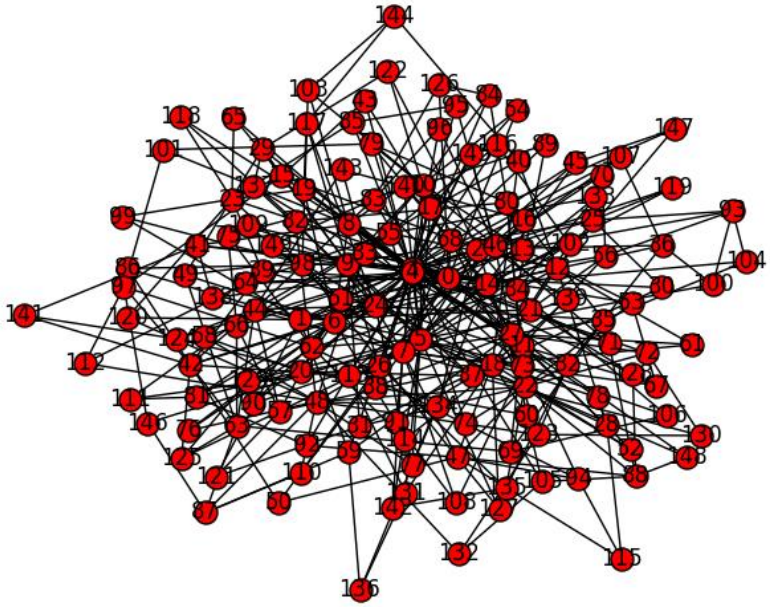


图 2-7 N=150 M=3 BA 拓扑

```
reqN and time =  400000 1463852175.2249746
reqN and time =  450000 1463852185.8931694
reqN and time =  500000 1463852196.5656118
req all import successfully!
Hit times Total  = 27607
ReqTimesTotal  = 500000
Hit Ratio= 0.055214
Hit times in the level 0 is : 10798  ratio: 0.021596
Hit times in the level 1 is : 9544  ratio: 0.019088
Hit times in the level 2 is : 6313  ratio: 0.012626
Hit times in the level 3 is : 948  ratio: 0.001896
Hit times in the level 4 is : 4  ratio: 8e-06
The average length of get: 3.650708
Stat End Successfully!
```

图 2-8 50 万组请求测试

此次测试我采用了 20 万，50 万与 100 万三组，每次命中率都略高于 5%，可以认为请求量已经足够。因此考虑时间成本我们统一拿 50 万组数据来标准化测试，标准化结果为：

数据测试时间为 196.565-090.020=106.545 秒

命中率 5.52%

平均长度 3.651

2.2.2 LRU 缓存替换策略

LRU 是最近最少使用页面置换算法(Least Recently Used),也就是首先淘汰最长时间未被使用的对象页面。在 ICN 网络内容节点中 LRU 就是把内容最旧的对象给舍弃，加入新的。考虑到 FIFO 先进先出的特点，我们可以使用线性的链表来实现，为了方便链表的操作，双向链表更加便捷。

插入方法加入链表的表头，删除最后一个,算法复杂度 O(1)。

查询方法也是利用 dict 来维护，接近于 O(1)复杂度。

标准化测试结果如下图

```
reqN and time =  500000 1463884549.9169939
Hit times Total  = 18752
ReqTimesTotal  = 500000
Hit Ratio= 0.037504
Hit times in the level 0 is : 7835  ratio: 0.01567
Hit times in the level 1 is : 6492  ratio: 0.012984
Hit times in the level 2 is : 3842  ratio: 0.007684
Hit times in the level 3 is : 579  ratio: 0.001158
Hit times in the level 4 is : 4  ratio: 8e-06
The average length of get: 3.6834
Stat End Successfully!
```

图 2-9 LRU 标准化测试

标准化结果为：

数据测试时间为 549.916-413.072=136.844 秒

命中率 3.75%

平均长度 3.683

2.2.3 Random 缓存替换策略

随机缓存替换（Random）是一种简单的替换策略，缓存池满时再加入新缓存块就需要先随机取出一个已缓存块。代码也比较简单，只需要在满时随机化取出一块即可，依旧可以用 dict 字典来维护。

实践中，小测试结果与 LFU 相同。

直接拿 50 万的标准化数据测试，有下图所示：

```
reqN and time =  400000 1463852751.9174435
reqN and time =  450000 1463852767.8620625
reqN and time =  500000 1463852783.651036
req all import successfully!
Hit times Total  = 18666
ReqTimesTotal  = 500000
Hit Ratio= 0.037332
Hit times in the level 0 is : 7583  ratio: 0.015166
Hit times in the level 1 is : 6486  ratio: 0.012972
Hit times in the level 2 is : 3944  ratio: 0.007888
Hit times in the level 3 is : 653  ratio: 0.001306
The average length of get: 3.68448
Stat End Successfully!
```

图 2-10 Random 标准化测试

此次 50 万组数据测试时间为 783.651-626.011=157.640 秒

命中率 3.73%

平均长度 3.684

2.2.4 CLOCK 缓存替换策略

CLOCK 是比 Second-chance 更加优化的 FIFO，因为页面不需要一直从链表最后入队，而是利用循环链表像钟表一样入队。迭代器每次都指向列表中最后被检查过的页面缓存。当页面错误发生时且无空帧（即缓存池满了且之前未缓存）后，被引用的对象在首部（迭代器指向的）被监测到。如果此对象为 0，那新的缓存对象则放在迭代器指向的位置。否则，此位置被清空，然后迭代器指向下一个，依次循环往复。

实现只需要加一个 self.hand 指向迭代器位置即可，小数据测试正确。

直接拿 50 万的标准化数据测试，有下图所示：

```
reqN and time =  450000 1463853163.8246047
reqN and time =  500000 1463853174.869752
req all import successfully!
Hit times Total  = 18965
ReqTimesTotal  = 500000
Hit Ratio= 0.03793
Hit times in the level 0 is : 7938  ratio: 0.015876
Hit times in the level 1 is : 6565  ratio: 0.01313
Hit times in the level 2 is : 3876  ratio: 0.007752
Hit times in the level 3 is : 582  ratio: 0.001164
Hit times in the level 4 is : 4  ratio: 8e-06
The average length of get: 3.68244
Stat End Successfully!
```

图 2-11 CLOCK 标准化测试

此次 50 万组数据测试时间为 174.869-065.305=109.564 秒

命中率 3.79%

平均长度 3.682

2.2.5 LIRS 缓存替换策略

LIRS(Low Inter-Reference Recency Set)算法针对 LRU 进行了性能上的优化，这是通过将重用距离作为动态排序决策的尺度来考虑而实现的。LIRS 维护一个变长 LRU 的 queue 队列，且此队列的 LRU 端为最近最少被访问过 2 次的第 Llirs 项(Llirs 为算法的参数)。

虽然对于 ICN 中大量视频 VoD 的 chunk 片段，LIRS 这种基于访问频率的策略可以缓存高流行度的内容片段块，提高了命中概率。但 VOD/视频直播的热点块可能随时间快速变化，而且此策略无法合理考虑视频内容块的顺序访问（即非独立参考模型），故 LIRS 不大适合 ICN 具体场景，无需仿真。

2.2.6 ARC 缓存替换策略

Adaptive Replacement Cache（简称 ARC）是自适应缓存替换算法，这个缓存算法是 LRU 与 LFU 的综合，由 2 个 LRU 组成。第一个，也就是 LRU1，包含条目都是最近只被用过一次的对象块，而第二个 LRU2 则乃被缓存后再次查询过的内容块。这也就是说 LRU1 缓存里是新对象，而 LRU2 放置的是常用的块。ARC 被评价为性能顶级的缓存算法，可以进行自适应性的调整，同时也是低负

载的。

实际设计时由于 List 对象设计是单向链表实现，从头插入 insert 内容的时间复杂度是 O（N），因此选用 deque 类型，时间复杂度 O（1）。

小数据测试与之前相同。

直接拿 50 万的标准化数据测试，有下图所示：

```
reqN and time =  450000 1463853566.306141
reqN and time =  500000 1463853577.7598264
req all import successfully!
Hit times Total  = 25464
ReqTimesTotal  = 500000
Hit Ratio= 0.050928
Hit times in the level 0 is : 8009  ratio: 0.016018
Hit times in the level 1 is : 11344  ratio: 0.022688
Hit times in the level 2 is : 5442  ratio: 0.010884
Hit times in the level 3 is : 666  ratio: 0.001332
Hit times in the level 4 is : 3  ratio: 6e-06
The average length of get: 3.660552
Stat End Successfully!
```

图 2-12

此次 50 万组数据测试时间为 577.759-460.145=117.614 秒

命中率 5.09%

平均长度 3.661

2.3 本章小节

除了 LIRS 在讨论中被排除以外，其余策略均进行了 50 万组的标准化测试，结果如下表 2.1：

	LFU	LRU	RANDOM	CLOCK	ARC
HIT RATIO %	5.52	3.75	3.73	3.79	5.09
TIME/S	106.545	136.844	157.640	109.564	117.614
LENGTH	3.651	3.683	3.684	3.682	3.661

表 2-1 替换算法比较

综合上表，有平均请求距离各组相差很少（因为命中率都不超过 6%），但是 LFU 命中率最高且处理时间较少，LFU 可以被视为最佳替换算法，我们之后的工作基于 LFU。

之前缓存研究中的一些结论也被我们重新审视。LRU 在之前的传统研究中是性能较高的替换策略被广泛采用，但是在 ICN 场景下却并不突出。ARC 作为

LRU 的自适应改进版本在 ICN 场景下的效率的确高于 LRU，也可以考虑使用。Random 策略与 LRU 结果上相差不多，之前有学界观点认为 ICN 网络中缓存替换策略只需要简单易于实现，具体算法的实际效果区别都不大。但是我们的测试表明 ICN 中缓存的替换策略依然十分重要，LFU 相比于 RANDOM 这种所谓的易实现策略在命中率上多了约 50%，最终速度也更快。

第 3 章 ICN 中缓存决策策略

本章主要研究了在 ICN 网络中，缓存在路径上的决策策略对网络性能的影响。由于上章节已经发现 LFU 最适合 ICN 网络环境，因此本章节的仿真模拟皆基于 LFU 策略

3.1 缓存决策策略介绍

常见的缓存决策策略学界已多有研究[2]，如下图所示：

LCE(Leave Copy Everywhere)	在对象返回的所有节点保留缓存的拷贝
LCD(Leave Copy Down)	当缓存命中时，在命中节点的下游节点拷贝一份文件。这种缓存决策策略需要多次访问同一个文件才会将文件缓存到网络的边缘，可以说考虑了文件的访问频率
MCD(Move Copy Down)	缓存命中时，将文件缓存至命中节点的下游节点，同时在本节点删除该缓存
Prob(Copy with Probability)	访问返回的所有节点一概率 p 缓存目标文件，当 p 为 1 时即变成 LCE
RCOne	在对象返回的沿途中随机的选择一个节点来缓存对象
ProbCache	在对象返回的沿途中以与请求者距离的倒数为概率缓存对象，即离请求者越近的节点缓存对象的概率越高

图 3-1 缓存决策策略表

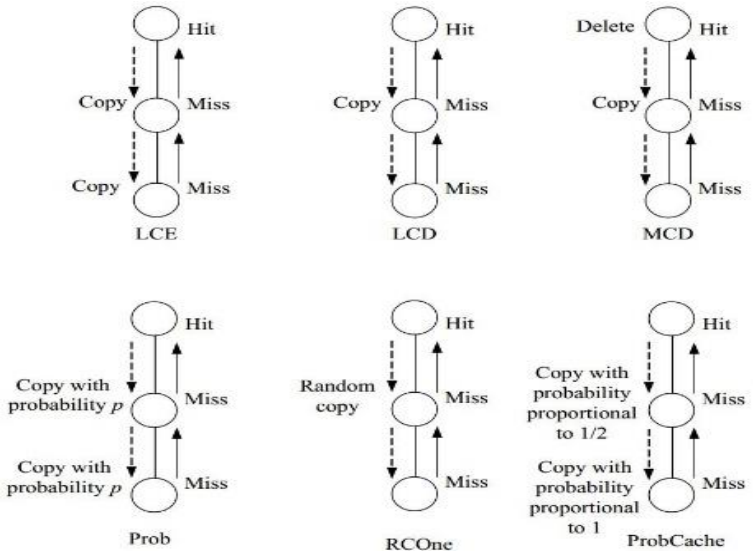


图 3-2 缓存决策图示

3.1.1 LCE 决策策略

本策略 LCE 与 LFU 的组合之前一章已经测过，50 万组请求标准化结果如下：

数据测试时间为 196.565-090.020=106.545 秒

命中率 5.52%

平均长度 3.651

3.1.2 LCD 决策策略

本策略实现与 LCE 类似，只是在得到数据后仅更新前一个节点。

利用 50 万组请求的标准化测试，有结果如下图：

```
reqN and time =  500000 1463881868.3341305
Hit times Total  = 31215
ReqTimesTotal  = 500000
Hit Ratio= 0.06243
Hit times in the level 0 is : 11337  ratio: 0.022674
Hit times in the level 1 is : 13439  ratio: 0.026878
Hit times in the level 2 is : 5635  ratio: 0.01127
Hit times in the level 3 is : 801  ratio: 0.001602
Hit times in the level 4 is : 3  ratio: 6e-06
The average length of get: 3.634006
```

图 3-3 LCD 标准化测试

数据测试时间为 868.334-784.192=84.140 秒

命中率 6.24%

平均长度 3.634

3.1.3 MCD 决策策略

由于 MCD 策略要删除请求节点的对应块，但是实际环境下 zipf 分布保障了前 20%流行度的块被请求次数过半，在 ICN 环境下只会因为删除了热门块而比 LCD 效率更低，命中下降，因此在 ICN 网络中没有实际价值，不如 LCD，不考虑额外测试。

3.1.4 Prob 决策策略

Prob 决策策略即每个路径上的节点都有 p 的可能性被替换，p=1 时退化为 LCE,p=0 时不缓存。这是两个极端情况，我们去测试 p 较大与 p 较小的两情况，分别取 p=0.2 与 p=0.8

p=0.2 如下图所示

```
reqN and time =  500000 1463881533.352136
Hit times Total  = 29051
ReqTimesTotal  = 500000
Hit Ratio= 0.058102
Hit times in the level 0 is : 10241  ratio: 0.020482
Hit times in the level 1 is : 11291  ratio: 0.022582
Hit times in the level 2 is : 6573  ratio: 0.013146
Hit times in the level 3 is : 943  ratio: 0.001886
Hit times in the level 4 is : 3  ratio: 6e-06
The average length of get: 3.64675
```

图 3-4 P=0.2

数据测试时间为 533.352-433.677=99.675 秒

命中率 5.81%

平均长度 3.647

p=0.8 如下图所示

```
req all import successfully!
reqN and time =  500000 1463881226.9675694
Hit times Total  = 30974
ReqTimesTotal  = 500000
Hit Ratio= 0.061948
Hit times in the level 0 is : 11425  ratio: 0.02285
Hit times in the level 1 is : 11541  ratio: 0.023082
Hit times in the level 2 is : 7022  ratio: 0.014044
Hit times in the level 3 is : 984  ratio: 0.001968
Hit times in the level 4 is : 2  ratio: 4e-06
The average length of get: 3.638398
```

图 3-5 P=0.8

数据测试时间为 226.967-144.190=82.777 秒

命中率 6.19%

平均长度 3.638

3.2 本章小结

这几种决策策略标准化测试结果如下:

LFU +	LCE	LCD	Prob-0.2	Prob-0.8
HIT RATIO%	5.52	6.24	5.81	6.19
TIME/s	106.545	84.140	99.675	82.777
LENGTH	3.651	3.634	3.647	3.638

总体而言，LCD 与 Prob-0.8 的效果比较理想，LCD 是最佳的缓存策略，无论是命中率还是总时间都优于其他策略。

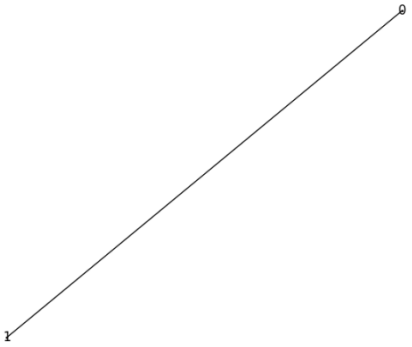
这与之前学界中 LCE 最优的观点相冲突。通过观察各级缓存的命中率，我发现 LCD 比其他策略在 0、1 级的命中率上要高 10%-20%，这也是实验中 LCD 性能较好的原因。这是因为虽然被请求资源热度分布符合 zipf，但是资源接入点与请求点并非那么集中，LCD 减少了对低热度缓存资源的冗余复制，使得在任何地点都热门的块或者在某一区域特别热门的块才被逐级复制缓存。

因此，在 ICN 网络中选择 LFU+LCD 是考虑性能后的最佳选择。

第 4 章 单节点大量数据

单节点，我们设置文件块有 3000 个，缓存大小 100 时，测量单节点。此时不受 Placement 策略的影响，仅测试替换算法性能。

拓扑如下，我们让所有请求从 0 进去向 1 请求，则 0 点的缓存命中率（即为 Cache level 0 命中数/总请求数）就是单节点缓存命中率。



4.1 LRU 测试

```
reqN and time =  0 1464188331.379825
req all import successfully!
reqN and time =  50000 1464188339.5069726
Hit times Total  = 5448
ReqTimesTotal  = 50000
Hit Ratio= 0.10896
Hit times in the level 0 is : 5448  ratio: 0.10896
The average length of get: 1.89104
```

5 万组节点时，测试结果如下

数据测试时间为 339.507-331.379=8.128 秒

命中率 10.896%

平均长度 1.891

其中命中率达 10%是因为块流行度呈现 zipf 分布，而非平均分布。如果平均分布命中率应该仅 100/3000=3.33%，这是合理的。

由于点数过少，速度过快，也加大到 50 万组测试对比

```
req all import successfully!
reqN and time =  500000 1464189058.3797371
Hit times Total  = 54502
ReqTimesTotal  = 500000
Hit Ratio= 0.109004
Hit times in the level 0 is : 54502  ratio: 0.109004
Total time speed (seconds)= 81.51562094688416
The average length of get: 1.890996
```

数据测试时间为 81.516 秒

命中率 10.900%

平均长度 1.891

几乎无偏差，因此之后都折中按 10 万组来进行标准化

下面是 10 万组的

```
req all import successfully!
reqN and time =  100000 1464190681.3453763
Hit times Total  = 10921
ReqTimesTotal  = 100000
Hit Ratio= 0.10921
Hit times in the level 0 is : 10921  ratio: 0.10921
Total time speed (seconds)= 16.248882293701172
The average length of get: 1.89079
```

数据测试时间为 16.249 秒

命中率 10.921%

平均长度 1.891

4.2 LFU 测试

```
req all import successfully!
reqN and time =  100000 1464190329.0452151
Hit times Total  = 13764
ReqTimesTotal  = 100000
Hit Ratio= 0.13764
Hit times in the level 0 is : 13764  ratio: 0.13764
Total time speed (seconds)= 13.555502891540527
The average length of get: 1.86236
```

数据测试时间为 13.556 秒

命中率 13.764%

平均长度 1.862

4.3 Random 测试

```
req all import successfully!
reqN and time =  100000 1464190769.1658826
Hit times Total  = 9776
ReqTimesTotal  = 100000
Hit Ratio= 0.09776
Hit times in the level 0 is : 9776  ratio: 0.09776
Total time speed (seconds)= 17.525293350219727
The average length of get: 1.90224
```

数据测试时间为 17.525 秒

命中率 9.776%

平均长度 1.902

4.4 CLOCK 测试

```
reqN and time =  100000 1464190841.1053643
Hit times Total  = 11760
ReqTimesTotal  = 100000
Hit Ratio= 0.1176
Hit times in the level 0 is : 11760  ratio: 0.1176
Total time speed (seconds)= 13.199438333511353
The average length of get: 1.8824
Stat End Successfully!
Press any key to continue . . .
```

数据测试时间为 13.199 秒

命中率 11.760%

平均长度 1.882

4.5 ARC 测试

```
reqN and time =  100000 1464190913.5715325
Hit times Total  = 19468
ReqTimesTotal  = 100000
Hit Ratio= 0.19468
Hit times in the level 0 is : 19468  ratio: 0.19468
Total time speed (seconds)= 13.699531316757202
The average length of get: 1.80532
```

数据测试时间为 13.700 秒

命中率 19.468%

平均长度 1.805

4.6 算法对比

五种策略均进行了 10 万组的标准化测试，结果如下表 4-1：

	LFU	LRU	RANDOM	CLOCK	ARC
HIT RATIO %	13.764	10.921	9.776	11.760	19.468
TIME/S	13.556	16.249	17.525	13.199	13.700
LENGTH	1.891	1.891	1.776	1.882	1.805

表 4-1 替换算法比较

综合上表，有平均请求距离各组相差很少，但是 ARC 命中率最高且处理时间较少，ARC 可以被视为单点最佳替换算法。

之前缓存研究中的一些结论也被我们重新审视。LRU 在之前的传统研究中是性能较高的替换策略被认为强于 LFU 被广泛采用，但是在 ICN 场景下却并不突出。

ARC 作为 LRU 的自适应改进版本在 ICN 场景下的效率的确高于 LRU，也可以考虑使用。

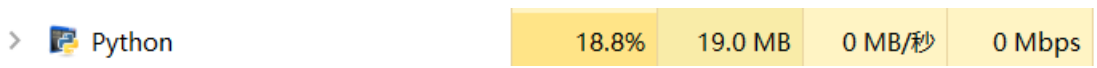
Random 策略结果上连 LRU 都不如，之前的学界对其在 ICN 网络中的观点大有问题。

我们的测试表明即使在单节点缓存中的替换策略依然十分重要，ARC 相比于 RANDOM 这种所谓的易实现策略在命中率上多了约 100%，最终速度也更快。

4.7 统计误差

由于单次测试可能存在统计上的误差与外部因素影响，我们需要对其统计误差进行估计。

Python 的设计机制通过 Global Interpreter Lock 全局解释器锁(GIL) 让同一时刻只能有一个线程对于 python 对象进行操作，而我们所用的仿真程序正是单线程的，因此每次运行最多把单 CPU 性能跑满，且都在 19%左右。



为了统计同一组数据准确性，我们对 ARC 的时间指标多次测试以统计标准差，我连续做了 8 组测试，分别为：

13.65272

13.66780

13.67920

13.68763

13.69368

13.69952

13.70273

13.70186

样本的均值 Mean：13.6856425

样本的标准差 Standard Derivation: 0.017967742

我们可以看到逐渐上升的趋势，这是由于反复实验中由于散热原因，虽然风扇开到最大依然使 CPU 温度升高超过 70 摄氏度，导致半导体的物理性能变化，计算能力下降，而本程序又是 CPU 计算密集型的，最终花费时间不断增加。但是到了最后三组开始稳定，实际上那时风扇温度已经温度在 75 度左右。

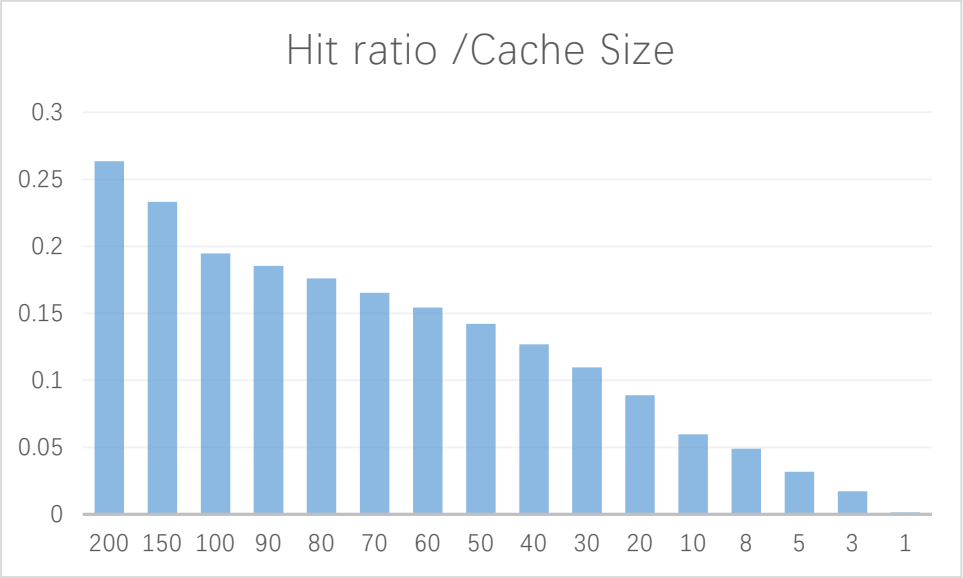
总的来说，即使是从未热机时到逐渐稳定影响伴随着时间数据的上升，但是误差也在 0.5%以内，与 Random LFU 之间 30%的差距比非常小，所以数据可信，可以对其进行比较。

4.8 缓存池大小影响

再以 ARC 为例子，研究缓存池大小对性能测试的影响，原先 replaceCacheSize 是 100，我将其从 200 逐渐降至 1 测试各种性能，结果如下：

	200	150	100	90	80	70	60	50	40	30	20	10	8	5	3	1
hit ratio	0.2636	0.23328	0.19468	0.18539	0.17615	0.16536	0.15445	0.14223	0.12679	0.10962	0.08887	0.05984	0.04895	0.03187	0.01731	0.00147
time	13.99157	13.79116	13.7									13.61303				

由于 Time 差别不大，与 13.700 秒偏差在 5%内，所以不是每组都贴了数据，但是 hit ratio 的差别很大，图像如下：



基本证明了缓存池大小对命中率影响很大，对总时间影响较小。

第 5 章 总结

ICN 网络要求网络在节点中缓存替换与路径上的决策时都能高速高效，所以本文测试了多种缓存决策算法、替换策略的缓存性能，并得出结论，LFU 在大吞吐量 ICN 网络中表现出色，其缓存性能与处理速度最优。

ICN 网络缓存系统在 LFU 替换策略确定后应该采取 LCD 的决策策略最优，而非一般学界所预设的 LCE 策略。

互联网的高速发展的同时架构上的弊端也开始逐渐体现。传统的点对点连接，TCP/IP 体系为核心的互联网架构体系在目前以内容为核心的互联网信息结构中越来越低效。新一代网络体系 ICN 是为了解决传统网络架构中的固有问题而设计的，而在新一代网络体系中，遍布核心网络和边缘节点的内置缓存是非常重要的环节。本文主要研究了在 ICN 网络中，缓存的替换策略和决策策略对于网络性能的影响。由于目前网络上的信息流量并非 ICN 架构下块级别数据，我不得不按照 ICN 的基本原则特征来模拟生成网络拓扑与块级别的大量数据，并依此进行仿真，通过标准化测试衡量不同的缓存替换策略以及缓存决策策略在 ICN 网络中的性能，并且得出了一些有用的结果，质疑了学界的某些预设观点。ICN 互联网架构的具体设计依然需要更多的研究，缓存在 ICN 中的作用也会更加重要，所以如何更加合理的利用缓存依然是值得研究的问题。

参考文献

[1] Index V N. Cisco Visual Networking Index: Forecast and Methodology, 2010--2015[J]. White Paper Cisco Systems Inc, 2011.

[2] Borst S, Gupta V, Walid A. Distributed caching algorithms for content distribution networks[C]// Conference on Information Communications. IEEE Press, 2010:1478-1486.

[3] Chin K W, Judge J T. Inter private network communications between IPv4 hosts using IPv6: US, US 7277453 B2[P]. 2007.

[4] Koponen T, Chawla M, Chun B G, et al. A data-oriented (and beyond) network architecture[J]. Acm Sigcomm Computer Communication Review, 2007, 37(4):181-192.

[5] Han L, Kang S S, In H P. An adaptive loss protection for video transmission over content-centric networks[J]. Multimedia Tools & Applications, 2014:1-14.

[6] Jacobson V, Burke J, Estrin D, et al. Named Data Networking (NDN) Project 2012 - 2013 Annual Report[J]. 2013.

[7] Aziziyeh A, Easty T, Khayat Z, et al. XIA: An Architecture for an Evolvable and Trustworthy Internet[J]. Proceedings of Acm Workshop on Hot Topics in Networks, 2011, 63(3):879-884.

[8] Ahlgren B, Dannewitz C, Imbrenda C, et al. A Survey of Information-Centric Networking (Draft)[J]. Communications Magazine IEEE, 2011, 50(7):26 - 36.

[9] Zahemszky A, Gajic B, Rothenberg C E, et al. Experimentally-Driven Research in Publish/Subscribe Information-Centric Inter-Networking[C]// Testbeds and Research Infrastructures. Development of Networks and Communities -, International ICST Conference, Tridentcom 2010, Berlin, Germany, May 18-20, 2010, Revised Selected Papers. 2010:469-485.

[10] Ahlgren B, Dannewitz C, Imbrenda C, et al. A Survey of Information-Centric Networking (Draft)[J]. Communications Magazine IEEE, 2011, 50(7):26 - 36.

[11] Paul S, Pan J, Jain R. Architectures for the Future Networks and the Next Generation. [J]. Computer Communications, 2011, 34(1):2-42.

[12] Choi J, Han J, Cho E, et al. A Survey on content-oriented networking for efficient content delivery[J]. IEEE Communications Magazine, 2011, 49(3):121-

127.

[13] Rahman A, Kutscher D, Song H, et al. DECADE: DECoupled Application Data Enroute[J]. 2013.

[14] Netze H. DECoupled Application Data Enroute (DECADE) Problem Statement[J]. Heise Zeitschriften Verlag, 2012.

[15] Ghodsi A, Koponen T, Rajahalme J, et al. Naming in content-oriented architectures[J]. Icn, 2011, 14(5):94-97.

[16] Guoqiang Z, Mingdong T, Suqi C, et al. P2P traffic optimization[J]. Sciece China Information Sciences, 2012, 55(7):1475-1492.

[17] Fricker C, Robert P, Roberts J, et al. Impact of traffic mix on caching performance in a content-centric network[J]. Proceedings - IEEE INFOCOM, 2012:310 - 315.

[18] Psaras I, Chai W K, Pavlou G. Probabilistic in-network caching for information-centric networks[C]// Edition of the Icn Workshop on Information-Centric NETWORKING. ACM, 2012:55-60.

[19] Perino D, Varvello M. A reality check for content centric networking[C]// Acm Sigcomm Workshop on Information-centric Networking. ACM, 2011:44-49.

[20] Wierzbicki A, Leibowitz N, Ripeanu M, et al. Cache replacement policies revisited: the case of P2P traffic[C]// IEEE International Symposium on CLUSTER Computing and the Grid. 2004:182-189.

[21] Hefeeda M, Saleh O. Traffic Modeling and Proportional Partial Caching for Peer-to-Peer Systems[J]. IEEE/ACM Transactions on Networking, 2008, 16(6):1447-1460.

[22] Zipf G K. Selected studies of the principle of relative frequency in language.[J]. Language, 1932, 9(1):89-92.

致谢

时间飞逝，大学的学习生活很快就要过去，在这四年的学习生活中，收获了很多，而取得这些成绩和一直关心帮助我的人分不开的。

首先非常感谢学校开设这个课题，为本人日后的学习提供了经验，奠定了基础。本次毕业设计是对我大学四年学习下来最好的检验。经过这次毕业设计，我的能力有了很大的提高，比如操作能力、分析问题的能力、合作精神、严谨的工作作风等方方面面都有很大的进步。这期间凝聚了很多人的心血，在此我表示由衷的感谢。没有他们的帮助，我将无法顺利完成这次设计。

首先，我要特别感谢我的指导老师徐跃东老师对我的悉心指导，在我的论文书写及设计过程中给了我大量的帮助和指导，为我理清了设计思路和操作方法，并对我所做的课题提出了有效的改进方案。徐跃东老师渊博的知识、严谨的作风和诲人不倦的态度给我留下了深刻的印象。从他身上，我学到了许多能受益终生的东西。再次对徐跃东老师表示衷心的感谢。

其次，我要感谢大学四年中所有的任课老师和辅导员在学习期间对我的严格要求，感谢他们对我学习上和生活上的帮助，使我了解了许多专业知识和为人的道理。

另外，我还要感谢大学四年和我一起走过的同学朋友对我的关心与理解，与他们一起学习、生活，让我在大学期间生活的很充实，给我留下了很多难忘的回忆。

最后，我要感谢我父母对我的支持，如果没有他们在我学习生涯中的奉献，我将无法顺利完成今天的学业。