**安徽大学**

**本科毕业论文（设计、创作）**

**题　　目：　　基于负载特征的Redis数据库性能预测**

**学生姓名：**　 　周元核 **学号：** E31714013

**院（系）： 计算机科学与技术学院 专业：　　网络工程**

**入学时间：　 2017　　　　年　　7　　　月**

**导师姓名：　　窦晖 　　 职称/学位：　副教授**

**导师所在单位：　计算机科学与技术学院**

**完成时间：　 2021　　　　年　　　5　　　　月**

**基于负载特征的Redis数据库性能预测**

**摘 要**

Redis键值型NoSql数据库，为互联网应用提供高效的解决方案，广泛应用于各种对性能要求较高的场景中。

Redis数据存储在内存当中，已有研究发现在数据量较大的时候，Redis性能会显著的受到操作数影响。例如：写入数据总量大于Redis的最大内存空间时会不断地触发写磁盘操作，进而影响到后续的数据读写性能。为了分析Redis在不同负载特征下读写性能，本文研究面向负载特征的Redis性能预测。

本文通过两台虚拟机分别部署Redis服务器和YCSB性能测试工具，利用YCSB的负载配置来模拟不同的读写请求。修改操作频率和操作类型、读写比例、数据随机和连续访问比例等要素模拟不同的负载特征，收集在不同的负载下Redis的吞吐量和延迟，作为性能指标数据，接下来采用多项式回归建模，并基于模型预测在给定读写请求下Redis的性能，为Redis的运行时性能优化提供理论依据。

关键词：Redis；负载特征；YCSB；性能预测；多项式回归

**Performance prediction of Redis based on workload characteristics**

**Abstract**

Redis key valued NoSQL database provides an efficient solution for Internet applications and is widely used in various scenarios with high performance requirements.

Redis data is stored in memory. Previous studies have found that when the amount of data is large, redis performance will be significantly affected by the number of operands. For example, when the total amount of data written is greater than the maximum memory space of redis, the write to disk operation will be triggered continuously, which will affect the subsequent data reading and writing performance. In order to analyze the read-write performance of redis under different load characteristics, this paper studies the performance prediction of redis based on load characteristics.

In this paper, the redis server and ycsb performance testing tool are deployed on two virtual machines, and the load configuration of ycsb is used to simulate different read-write requests. Modify the operation frequency and operation type, read-write ratio, data random and continuous access ratio and other factors to simulate different load characteristics, collect the throughput and delay of redis under different loads as the performance index data, then use polynomial regression modeling, and predict the performance of redis under a given read-write request based on the model, It provides a theoretical basis for the runtime performance optimization of redis.

**Keywords: Redis; load characteristics; YCSB; performance prediction;** **Polynomial Regression**

目 录

[目 录 4](#_Toc72158873)

[1. 引言 5](#_Toc72158874)

[1.1 Redis应用广泛 5](#_Toc72158875)

[1.2 Redis性能至关重要 5](#_Toc72158876)

[1.3 Redis性能受到负载特征影响 6](#_Toc72158877)

[1.4 研究内容 7](#_Toc72158878)

[2 研究背景与相关工作 7](#_Toc72158879)

[2.1 非关系数据库（NoSQL）与Redis数据库 8](#_Toc72158880)

[2.2 负载特征与YCSB 8](#_Toc72158881)

[2.3 Redis优化的相关工作 9](#_Toc72158882)

[3 基于负载特征的性能测试流程 10](#_Toc72158883)

[3.1 硬件配置 10](#_Toc72158884)

[3.2 软件配置 10](#_Toc72158885)

[3.2.1 配置Redis 10](#_Toc72158886)

[3.2.2 配置YCSB 11](#_Toc72158887)

[3.3 负载类型选择 11](#_Toc72158888)

[3.4 数据收集 12](#_Toc72158889)

[3.4.1 单个测试流程 12](#_Toc72158890)

[3.4.2 测试脚本编写与数据收集 13](#_Toc72158891)

[4 建模与验证 14](#_Toc72158892)

[4.1 数据预处理与可视化 14](#_Toc72158893)

[4.2 数据预测模型建立 15](#_Toc72158894)

[5 总结与体会 19](#_Toc72158895)

[参考文献 20](#_Toc72158896)

[附录A 负载特征性能测试 22](#_Toc72158897)

[附录B 测试脚本代码 25](#_Toc72158898)

[致谢 26](#_Toc72158899)

# 引言

## 1.1 Redis相关应用

在当前环境下，以大数据为代表的数字经济不断优化转型，依据逐渐成为了现在互联网上数字产业最主要的一批产业，其对数据高效存储的需求也日益提高，Redis就是在这样环境下，的一种高效的键值存储系统，得益于其数据存储的高吞吐量和高稳定性，在目前网络环境下Redis应用愈发广泛。

Redis在国内外网站中已被广泛作为缓存使用，例如新浪和GitHub使用Redis的缓存服务，多网站并行模式中Redis作为首选缓存数据库使用[1]。很多公司服务器采用Redis作为高速缓存的解决方案，Redis的地位近年来逐渐升高，尤其是在持久数据和状态数据存储等方面，使用率上有逐渐取代Mecahed成为首选的服务器端缓存组件的趋势。

Redis是一种内存键值型数据库，数据存储在内存当中，因此读写速度非常快，采用Redis作为解决方案的服务器压力承受能力相对较高。很多电商网站有秒杀和抢购的功能，在短时间内用户下单速度非常快，因此服务器必须使用高效的机制，如果服务器采用Redis服务器，整点库存发布时，库存信息存储在Redis中，凭借Redis的强大的读写能力减少用户对重新发现服务器的压力，减少短期过载的发生。

Redis支持网络，可用于数据库间通讯。对于远程、异步通讯时，如双方服务器均采用Redis进行数据缓存，则可以直接进行网络数据传输，因此可以在一定程度上替代传统的web服务或HTTP，实现高效的服务间通信；Redis和Logstash共同解决了分布式日志系统的问题，日志由Logstash代理收集根据服务器协议汇总成栈并发送给其中一个数据库，由Redis数据库重新发现，另一个中央存储库读取Redis的日志进行集中处理。

此外Redis还具有高度适应性，仅在JAVA环境下就有Jedis、Redisson、Lettuce等多种框架可用于开发，Redis充分吸收框架的优点和便利之处，采用分布式和可扩展的数据结构，可进行短期内存和持久日志内存[2]。

## 1.2 Redis具有高性能的特征

相比于传统的连接型数据库，其存数据存放在内存中而不是硬盘中，读写、查询、修改、删除等操作都是直接作用于内存，因此Redis和传统数据库速度差距就好比硬盘和内存的读写速度差距，由于Redis数据的高读写效率，Redis服务器面向的客户大多都有需要较高吞吐量的场景，在作为缓存、通讯、日志等功能使用的时候，尤其明显，例如，在2019年的前12分钟里中国某电商平台双十一成交总数为500亿笔，总成交额为2600亿。这类场景这要求数据库具有高的吞吐量和并发性， Redis性能对服务器的能否正常运转至关重要。

Redis本身对大量数据处理进行了很多优化，例如：可以通过线程优化数据资源调用，Redis将数据缓存到内存时会，为了保证执行效率，会将其分片处理。因此Redis作为缓存使用时候也具有优秀的读写效率，其高速读写的特性来自于数据存储在内存中，但是同时也容易受到内存或CPU读写能力限制。

Redis在提升效率的同时尽可能减少数据可靠性的牺牲，即使Redis存储时的键值列表采用的HashMap具有线程性，Redis本身的数据却依然是安全的，其原因在于操作的原子性，原子性也使得Redis具有可扩展性和分布式[3]。Redis数据持久化默认采用的是RDB方案，数据zai 指定的时间间隔后保存数据快照到内存中，而对数据可靠性要求更高的场景则提供了相对来说更加安全的AOF方案，在命令写入到操作日志后，同时会保存所有的历史操作

## 1.3 Redis性能会受到负载特征影响

在Redis的应用场景里，普遍需要Redis保持较高的读写能力,而Redis的读写能力很容易受到负载特征的影响，负载特征一词有多种解释，本次实验中的负载特征，如无特殊说明，均指的是在数据库性能测试中影响性能指标的因素此含义，负载特征可以由来源归纳为CPU型负载特征、内存型负载特征、文件系统型负载特征、磁盘型负载特征、网络负载特征[7]，正确选择负载类型和基准进行测试是模拟负载、测试数据库性能中里重要的步骤。

影响Redis服务性能最关键的一部分是服务器的内存使用。如果Redis服务器在运行过程中访问请求的内存超过了可使用的最大内存（通常来说，这必然会发生），或者内存使用最大内存，则Redis数据库必须更换内存和swap空间，将内存的旧或不使用的内存写入硬盘。此过程由操作系统实现，由于硬盘数据传输速度远小于内存的读写速度，因此当在Redis过程中发生内存交换时，Redis和依赖Redis上的数据应用程序将会受到性能影响。而在不同负载条件下，内存交换的频率会显著不同，因此在不同负载条件下Redis的性能也不尽相同，因此Redis在不同负载特征下对服务器的压力也不尽相同，为了能针对性的合理预测服务器性能，研究Redis性能受到负载特征的影响很有意义。

表格 1 负载特征分类

|  |  |  |
| --- | --- | --- |
| 名称 | 基本属性 | 实例 |
| CPU密集型 | 平均负载  用户时间与系统时间之比  系统调用频率  上下文切换频率  中断频率 | 在大多数公司里，服务器的负载有明显的时间趋势，其取决于活跃客户端的数量。多数游戏公司的服务器傍晚或节假日活跃用户数会比平均值高得多，这是一中CPU密集型的负载，每秒会执行高频率的系统调用和上下文切换。 |
| 内存密集型 | 系统范围的物理和虚拟内存使用率  饱和度：换页，交换，OOM终结者  内核和文件系统缓存使用情况  每个进程的物理内存使用情况  是否存在内存资源的控制 | 若某系统有256GB主存，系统其使用率只有1%。而主存的30%是文件系统缓存。那么对于此系统承受的负载特征就属于内存密集型负载 |
| 磁盘密集型 | I/O频率  I/O吞吐量  I/O大小  随机和连续比例  读写比 | 典型的磁盘密集型负载出现在传统电商数据库里，系统磁盘中的操作96%是读。一些秒杀活动里偶尔有小段的连续写爆发，可以把磁盘最大推到平时的数百倍。 |
| 网络密集型 | 网络接口吞吐量：RX和TX，B/s  网络接口IOPS：RX和TX，帧每秒  TCP连接率：主动和被动，每秒连接数 | 网络密集型负载其影响因素吞吐量随用户而变化并且变化，通常来说在正常服务器使用中很少有专业服务器因网络密集型负载瘫痪，多高强度的网络密集型负载来自网络攻击如DDos攻击等。 |

## 1.4 研究内容

本次实验根据Redis数据库特性，选择CPU密集型负载和内存密集型负载对Redis数据库进行性能测试，并根据操作频率和操作类型、读写比例、数据随机和连续访问比例等因素进行数据库性能测试。本文通过基于负载特征对Redis性能测试实验；对在不同负载特性下Redis服务器吞吐量和时延等性能指标信息，选择多项式模型，对不同负载情况下数据库性能进行建模。为数据库进行性能评估。在实际测试后验证了该模型进行负载预测时具有较高的准确性。

# 2 研究背景与相关工作

当前各大互联网公司在提供web服务时，有众多基于Redis的数据存储服务器，Redis支持的多种类型的数据结构刚好满足了源数据类型多样的存储需求，分析Redis服务器的负载能力，为Redis的运行时性能优化提供依据，对数据库有着重要意义。

## 2.1 Redis数据库

NoSQL是一种区别于关系型数据库，为了适应互联网进程而提出的全新的数据库类型，其跳出传统的行列关系，采用哈希表进行存储和查询，虽然服务器部署并不像传统数据库一样简单快捷，但是在面向局部数据查询修改时[5]，通常其擅长领域比传统关系型数据库有显著的不同。市面上NoSQL数据库类型很多，常见的有HBase、MongoDB、Redis等，虽然其数据存储方式各有差异，性能和应用场景也不尽相同，但是它们都有一个相同的特点，就是数据存储不需要固定的模式，在数据库进行横向扩展时非常迅速快捷。不仅仅是能过够快速为元素添加或删除属性，更是从架构的层面上使得属性具有可扩展性，使得属性不再受到元素、类型约束。尤其是在大数据量、高操作量的情况下，NoSQL数据库都具有非常优秀的读写性能，尤其在大数据量下，相对传统连接型数据库来说表现更加优秀。这得益于它的无关系性，创建数据库的结构简单。

Redis数据处理能力优秀，Redis在不同平台下有相同的指令操作，针对相同数据类型的也可以采取不同的指令处理方式， Redis的多线程技术和Java采用相同思想，这是Redis的主要特点之一，多线程处理，每线程优化系统资源，基于线程存储查找修改删除等方式对Redis数据库里的数据进行操作[6]。在数据片的存入硬盘和读取时会根据实际情况有选择的线程，Redis的最大优点是可以将数据缓存在内存中，并将数据存储在分区中，具有较高的读写效率。通过数据分片可以有效提高Redis的使用效率，加快数据采集速度。

Redis在部分场景里也经常被用作“数据结构服务器”，Redis的内存储的数据值类型非常自由，不仅可以是数字、字符串，也可以为映射、列表、集合和排序等集合。它还提供了基本的数据结构列表、集合、映射，大大提高了程序的效率。语言方面上Redis也非常广泛，支持C++、Java、.Net和许多其他语言调用[6]。一般来说，Redis受到消息队列的限制形式存在，作为内嵌的List存在，满足实时的高并发需求。而通常在一个电商类型的数据处理过程之中，有关商品，热销，推荐排序的队列，通常存放在Redis之中，期间也包扩Storm对于Redis列表的读取和更新。

Redis的优点也是其缺点，由于Redis数据存储在内存中，数据库容量受到物理内存的限制，在用作海量数据的高性能读写时会遇到较大的限制。 因此，研究在物理内存确定的情况下，Redis数据库读写能力随负载类型的变化，对Redis的优性能化很有意义。

## 2.2 负载特征

Redis一般需要保持较高的读写能力，但是Redis的读写能力很容易受到负荷特性的影响。负载特征由源总结为CPU型密集型、内存密集型、磁盘型密集型、网络密集型[7]，如果Redis的访问请求的内存超过了最大可用内存或者内存可以使用最大内存，那么必须对数据库进行数据交换，在内存和硬盘之间进行信息读写，将内存中旧的或未使用的东西写入硬盘。这个过程是由操作系统实现的。在更换内存和硬盘的侧面后，会留下新物理内存的新页或活动页，由于硬盘的读写速度远低于内存，因此在重新发现、重新发现和数据应用程序的过程中，内存的交换。

本次实验选用的性能测试软件为YCSB，是一款由Yahoo公司开发的开源软件，主要用来对在线以及云服务数据库进行数据测试的文件，既可以进行读写测试，也能用来随机数据生成。本次实验研究的负载特性对测试数据的要求是非常高的。本次实验选择“文件系统型负载特征”对Redis数据库进行性能测试，并根据操作频率和操作类型、读写比例、文件I/O大小、文件I/O吞吐量、读写位置分布和连续访问比例等要素进行数据库性能测试[8]。软件的预设负载模型可以分为模拟重写型负载、模拟重读型负载、模拟只读型负载、模拟查找型负载、模拟短区间查找型负载、模拟修改写回型负载六种。

## 2.3 Redis优化的相关工作

在已有的研究内容中，我们了解到内存的使用是Redis服务最关键的部分，在相同的容量时，内存的市场价格远高于硬盘，因此即使在专业服务器中也经常出现内存难以完全存储数据的情况，如何充分利用内存、使得物理内存能充分发挥作用，抑或是服务器在承受不同压力时需要分配的内存、算力是Redis成本降低的研究关键[9]。研究已知Redis服务器中的内存使用率超过了数据存储上限，会频繁触发内存写入的现象，系统交换硬盘和内存的空间，进行数据页面写入和重新插入。在Redis里，采用了 AOF、RDB两种数据持久化存储的机制。用于断电或和硬盘中数据交换时数据库内数据的恢复，Redis的数据库现场恢复等。

虽然在已有的研究中对Redis的性能受到什么因素影响分析已经很充分，但是关于数据库具体受到影响的具体范围仍然模糊，即使是相同规模的数据触发内存重写机制的概率也不尽相同，在自然语言语料库里，某个数据的出现的频率越高，其超过其它数据的比例也就越高。在自然语言语料库里出现频率最高的单词“the”出现的频率约为7%，出现频率第二位的单词“of”只有大约3.5%，只有第一名出现频率的一半不到，第三名往后出现的概率更低[10]，这类数据分布特征被称为Zipfian分布，在这种分布特征下，大约95%的数据访问会集中在5%的数据上，这种数据分布也是本次实验测试的重点数据分布类型。

本次实验研究的重点在于探究Redis在不同的负载类型下受到的影响，尤其是指对数据库的操作频率和对数据操作类型、数据读写比例、数据大小、随机和连续访问比例等要素下数据库性能的影响。通过实验，以吞吐量和平均时延作为性能直白哦。以多项式回归进行建模，得到了Redis数据库的负载特征模型，并在模型测试中对所得模型进行预测实验验证模型的可执行性，得到了关于负载特征的性能预测数据模型。

# 3 基于负载特征的性能测试流程

## 3.1 硬件配置

实验物理环境为学生个人桌面电脑，处理器为AMD R5-2600，CPU主频为3.85GHz, 主机内存为32G，内存频率为2400MHz. 物理环境大于实验分配的硬盘和内存空间，能满足系统的正常稳定运行，在本次实验中的数据测量里，多次获得的数据整体较为稳定，相对更具有说服力。

从VMware官网[11]下载并安装VMware Station，在软件中创建两个虚拟机，在两个虚拟机里安装ubuntu18.04操作系统。在其中一个虚拟机中安装Redis作为服务器数据库，在另一个虚拟机里安装YCSB作为进行数据测试的客户端，通过VMware Station软件内部设置配置两虚拟机的网络互通，进行数据测试。

表格 2 实验物理机器环境

|  |  |  |
| --- | --- | --- |
| 组件 | 型号 | 主频/内存 |
| CPU | R5-2600 | 3.85GHz |
| 内存 | 海盗船 | 32G |
| 虚拟服务器和客户机系统 | ubuntu-18.04 | - |
| 服务器内存分配 | - | 4G |
| 客户端内存分配 | - | 4G |

## 3.2 软件配置

### 3.2.1 配置Redis

在Redis官网[12]下载Redis（版本号为5.0.1）至服务器，解压缩。在文件目录下，执行Redis-server开启服务器，生成配置文件。在文件安装目录下打开配置文件Redis.conf，注释“ bind 127.0.0.1”并设置密码，重新启动服务器，开放访问来源UP，方便进行下一步测试。

Redis内置读写保护程序，在服务器被访问创建一个新的名为dump rdb的后台进程，若此进程创建时出现内存充满的情况，则会认为服务器被恶意数据攻击，停止访问，no则表示关闭该功能，由于本次实验进行基于负载特征的数据库运行能力预测。在测试过程中必然会遇到内存充满的情况，因此需要关闭该功能。

表格 3 Redis配置相关指令

|  |  |
| --- | --- |
| 指令 | 含义 |
| stop–writes–on–bgsave-error | Redis内置读写保护程序 |
| vm-enabled | Redis是否开启VM功能 |
| maxmemory | Redis最大内存使用率 |
| save t n | 每当n条指令被修改后，t秒后刷新一次 |
| port | Redis默认端口号为6379 |
| appendfsync | 刷新日志到dis的规则，默认为always |

### 3.2.2 配置YCSB

从官网[13][下载YCSB](https://github.com/brianfrankcooper/YCSB/releases/tag/0.10.0下载YCSB)数据包（版本号0.15.0）。将文件通过VMware-tools导入到客户端中，为了能正确识别服务器的Redis数据库，需要在在工作负载配置中，添加Redis的服务器地址和端口号，YCSB的数据库配置和负载类型在同位置，在安装目录下找到对应的工作负载workload，

表格 4 YCSB测试中负载特征的主要参数

|  |  |
| --- | --- |
| recordcount | load和run操作中，使用的YCSB实例记录数 |
| operationcount | load和run操作中，使用的YCSB实例操作数 |
| readproportion | 进行read的操作占所有操作的比例 |
| updateproportion | 进行update的操作占所有操作的比例 |
| insertproportion | 进行insert的操作占所有操作的比例 |
| scanproportion | 进行scan的操作占所有操作的比例 |
| requestdistribution | 使用什么分布来选择要操作的记录 |

## 3.3 负载类型选择

本次实验中，使用的负载类型为YCSB默认的负载类型中的五种，其中包含了模拟重写型负载、模拟重读型负载、模拟只读型负载、模拟查找型负载、模拟短区间查找型负载，为了模拟真实的数据访问情况，本次实验中多数的数据访问的对象会集中在少量的数据上，因此记录的分布选择为Zipfian分布。

表格 5 本次实验选用的负载类型和设置

|  |  |  |
| --- | --- | --- |
| 负载类型 | 负载名称 | 负载特征 |
| workloada | 读写均衡型负载 | 模拟读写平均型负载。对数据库的操作中读取和写入操作各占50%，操作的记录的分布为Zipfian分布。 |
| workloadb | 重读型负载 | 数据库的操作中读操作占95%，写操作占5%，操作的记录的分布为Zipfian分布以，是读为主要工作负载测试的参数设置。 |
| workloadc | 只读型负载 | 所有操作均为读操作，操作的记录的分布为Zipfian分布，是以读为全部工作负载测试的参数设置。 |
| workloadd | 查找型负载 | 读操作占95%，插入操作占5% ，操作的记录的分布为最近分布。 |
| workloade | 短区间负载 | 查找操作占95%，插入操作占5%，操作的记录的分布为Zipfian分布，且限制单次最大查找长度为100，是针对短区间生成的测试工作负载。 |

## 3.4 数据收集

## 3.4.1 单个测试流程

为了得到负载和数据库性能的指标，需要修改负载类型、数据量和操作数，进行测试得到数据模型。本次测试需要获得的数据有：吞吐量、单指令平均时延。

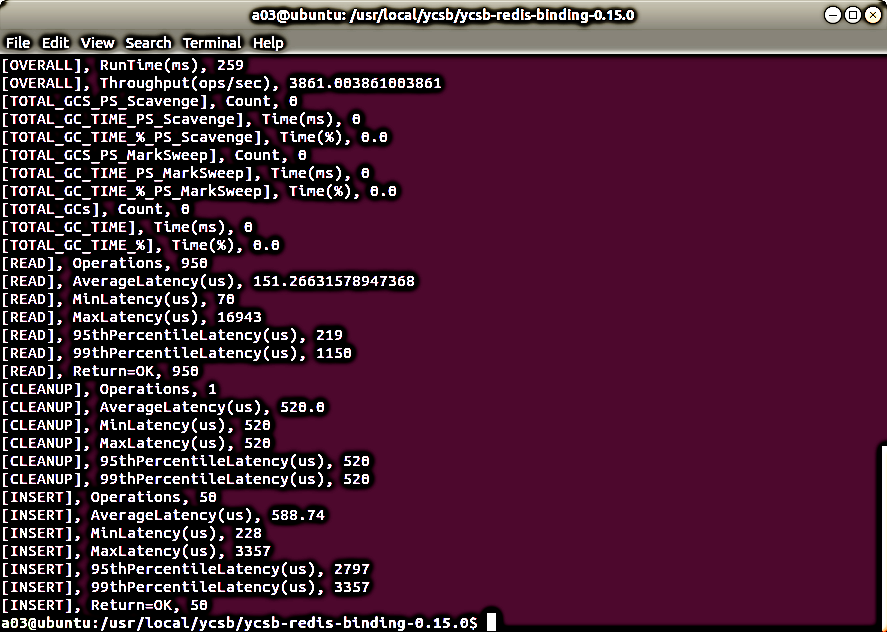


图例 1 单次测试流程

数据测试，单个数据测试流程分三步。

1.导入对应的工作负载，指令为load Redis -s -P workloads/workloada，导入数据库的初始数据。-P参数用来加载property文件。这个例子中，我用来加载workloada中的参数文件。负载类型来自workloada中预先的设置，文件中 recordcount意思是在Redis中的预先导入的数据数，每一个Key都是hash类型，我们可以重执行上面的命令，Redis指的是服务器类型。如不填写，默认为dummy客户端，不操作数据库，而将YCSB对数据库的操作输出到控制台。可以实用这个特性查看各种负载是如何执行到服务器中的。-s 参数表示要求Client向stderr输出状态报告。

2.使用工作负载进行测试，指令为run Redis -s -P workloads/workloada。和导入阶段主要的不同是，告诉服务器这一段指令意味着导入数据已结束，可以直接执行。



图例 2 单次性能测试结果展示

3.将得到的数据导出到OUTPUT.txt文件中。

## 3.4.2 测试脚本编写与数据收集

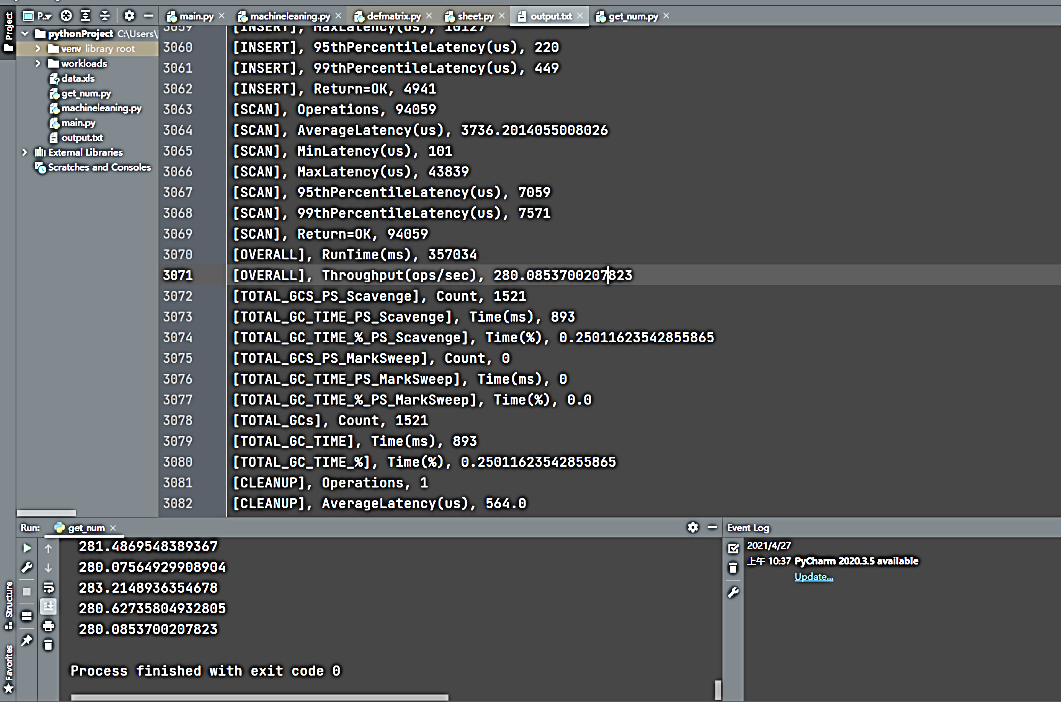
拟测试的数据量从10,000到100,000，以50000为步长。每组数据测试五次并取均值，由人工测量时间空间误差较大，易受物理环境变化影响，因此编写python脚本进行测试。



图例 3 脚本测试流程

脚本文件在客户机中运行，从文件中载入预先输入的工作负载类型,修改初始数据数操作数总数。将工作负载写入到workload中，随后执行单次工作负载的流程，得到的数据输入到OUTPUT.txt中，重复上述过程，直到从一千到十万数据测试完成。此时OUTPUT.txt中得到的就是当前工作负载的全部测试数据。第二步、修改工作负载类型，重复上述过程，直到全部工作负载均测试完成。

得到的数据文件中包含许多不必要信息，我们只需要提取出其中的吞吐量和运行时间等数据。通过正则表达式可以快速提出。



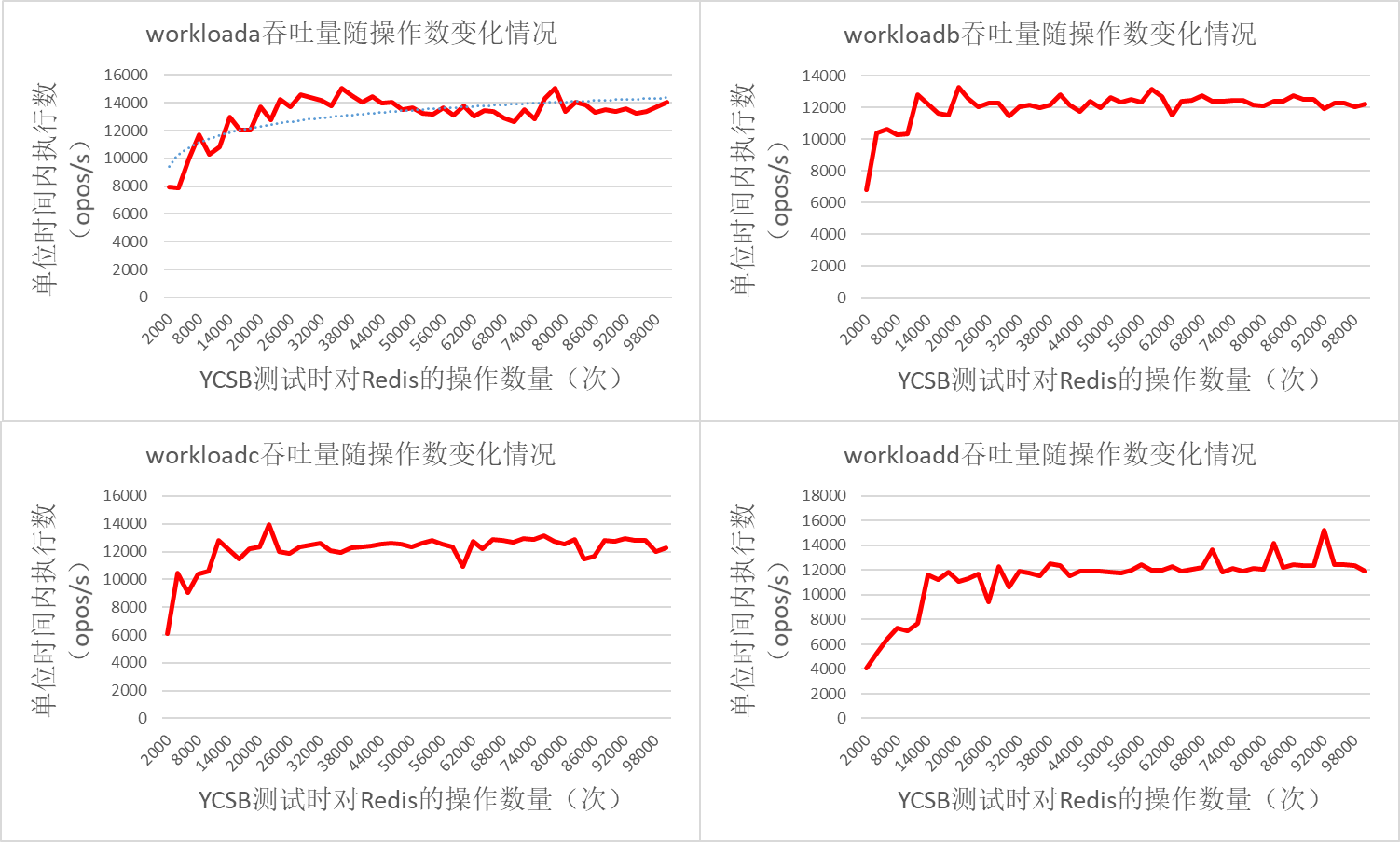
图例 4 workloada性能测试指标

# 4 建模与验证

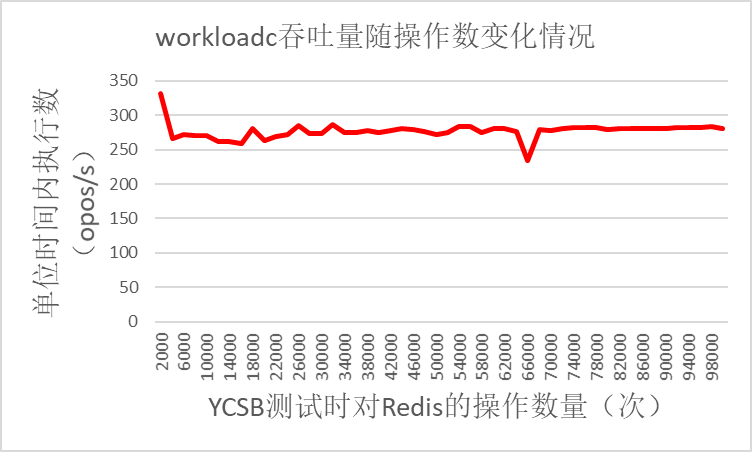
## 4.1 数据预处理与可视化

将实验得到的数据，绘制成曲线。可以看出，workloada ，workloadb ，workloadc ，workloadd四种负载下，单位内执行时间随数据量增大而增加，而且存在不同的上限。总执行时间（延时）则近似呈现正比例关系。对于workloade，由于服务器的处理能力受到查询的长度的限制，总吞吐量很低且则稳定。

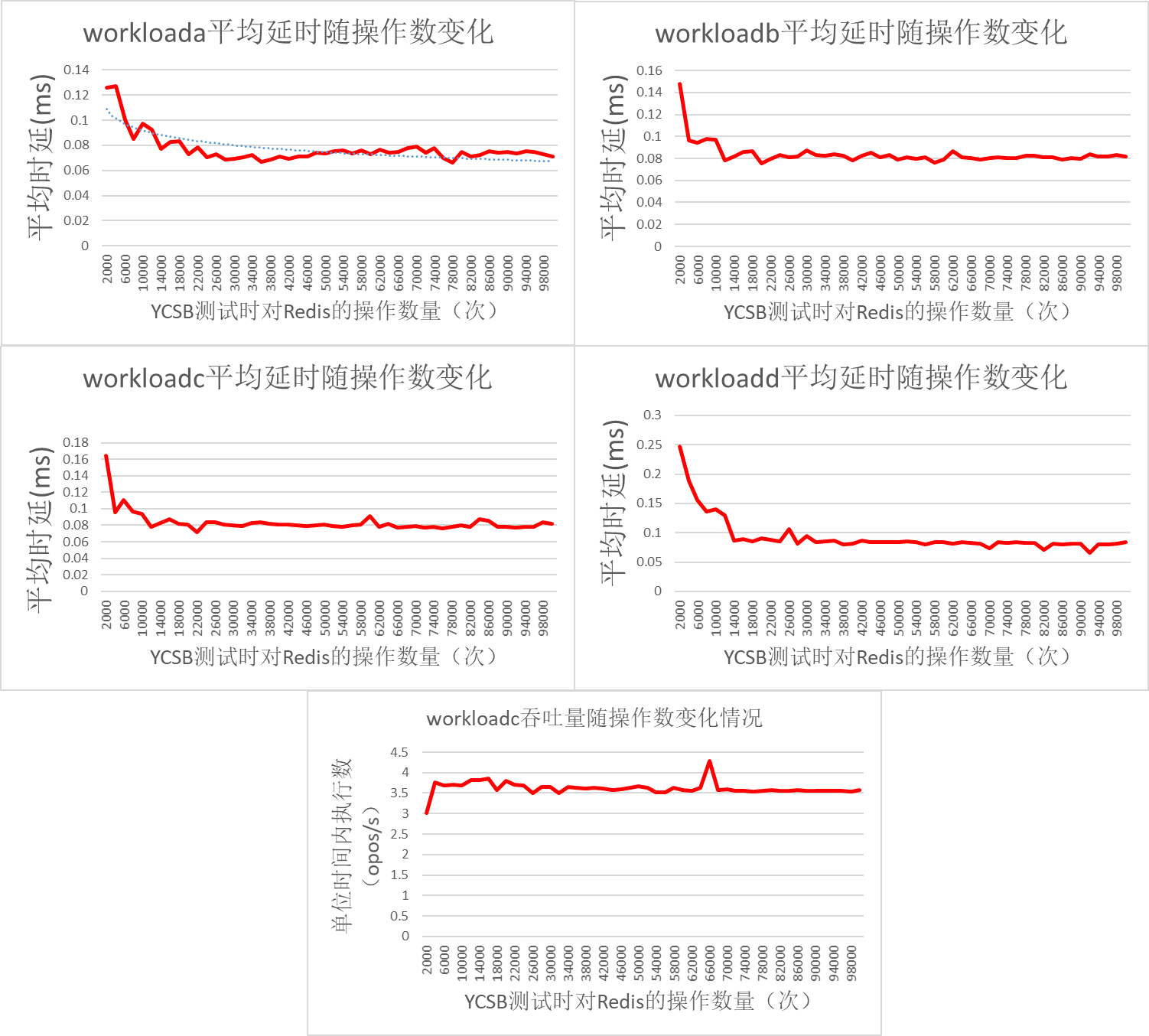
平均时延随着吞吐量增加而降低，且存在下限，



图例 5 workloada-workloadd 吞吐量随指令数变化情况



图例 6 workloade吞吐量随操作数(条)变化情况



图例 7 workloada-workloade 平均时延(ms)随操作数(条)变化情况

## 4.2 数据预测模型建立

分析数据图像可知，在A,B,C,D四种负载特征下，吞吐量随数据变化的图像成非线性关系，数据大多集中在一条满足函数关系的曲线上，采用多项式回归建模，修改多项式最高次数，依次测试回归结果， workload-A的回归模型测试结果如下表格所示：

表格 6 workloada 对性能指标进行多项式回归建模过程

|  |  |
| --- | --- |
| 拟合函数 | workloada吞吐量随访问数变化 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

据函数图像分析，Redis在workloada负载条件下，吞吐量随操作数变化的最适合选用的多项式次数为五次。每条指令的平均延时（ms）数学模型的多项式最高次数选择为六次。

**Workloada 吞吐量(ops/sec):**

(1)

**Workloada 单位时延(ms):**

(2)

据函数散点分析，Redis在Workload-B负载条件下，吞吐量随操作数变化的最适合选用的多项式次数为七次。每条指令的平均延时（ms）数学模型的多项式最高次数选择为七次。Workload上限较为平滑，在操作数为25000时达到上限，约为12300 ops/sec。平均实验与吞吐量近似成反比例关系，

**Workloadb 吞吐量(****ops/sec):**

(3)

**Workloadb 单位时延(ms):**

(4)

WorkloadC-E的数据建模过程大致相同，因此在此处仅展示结果，其它欠拟合和过拟合结果不再赘述。

**Workload-C 吞吐量(ops/sec):**

(5)

**Workload-C 单位时延(ms):**

(6)

**Workload-D 吞吐量(ops/sec):**

(7)

**Workload-D 单位时延(ms):**

(8)

**Workload-E 吞吐量(ops/sec):**

(9)

**Workload-E 单位时延(ms):**

(10)

表格 7 WorkloadC-E 对性能指标进行贝叶斯多项式建模结果

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 负载类型 | 吞吐量 | 负载类型 | | 平均单位时延 |
| c |  | c |  | |
| d |  | d |  | |
| e |  | e |  | |

# 5 总结与体会

在本次实验中，我进行了基于负载特征的Redis性能预测任务，经过复杂且繁琐的实验，我得到了Redis关于数据的变化规律，并将其绘制成图表，得到变化规律函数。这同时也是我第一次参与数据库类型的实验，实验前期可谓是难以下手，特别是实验环境在linux下，我本人在实验前完全没有接触过linux。即使是复制、粘贴、解压缩、运行可执行程序这样的基础操作也是一团雾水。一开始请教老师和学长的时候也感觉问题过于基础而感到难以启齿，搞得实验进度迟迟不能往前推进，老师在了解到我的情况之后，主动帮我联系学长，手把手从基础操作开始教我实验过程，在学长的建议下，我学习了linux系统下的各种基础操作，以及文件存储格式。经过数天的努力，成功的在linux系统上安装了Redis数据库和YCSB测试软件，并配置网络通讯。

本次实验对于我来说不仅仅是数据库测试实验，同时也是一扇新的计算机学科的大门，带我见识到了除了高级语言以外计算机专业的其它领域知识技能，感受到了由操作系统不同所带来的软件运行效率显著不同，以及不同平台下，程序适配也有显著的不同。今年我国自主研发的操作系统鸿蒙系统以及进入了公测阶段，到写此论文时间，我已经在网上报名了鸿蒙系统的试用，在接下来的学习中，我会积极学习在我国自主研发的系统下的编程，通过在不同系统下的编程多角度、全方面的认识和学习计算机这一学科。

参考文献

[1] 宁方美，贺雪梅，牟晋娟，Spring Boot集成Redis缓存技术在企业一卡通系统中的应用[J].电子技术与软件工程，2019(24):133-134.

[2] Kanellis， Konstantinos， Ramnatthan Alagappan， and Shivaram Venkataraman. "Too many knobs to tune? towards faster database tuning by pre-selecting important knobs." In 12th {USENIX} Workshop on Hot Topics in Storage and File Systems (HotStorage 20). 2020.

[3] Cao， Zhichao， Siying Dong， Sagar Vemuri， and David HC Du. "Characterizing， modeling， and benchmarking RocksDB Key-value workloads at Facebook." In 18th {USENIX} Conference on File and Storage Technologies ({FAST} 20)， pp. 209-223. 2020.

[4] Cooper， Brian F.， Adam Silberstein， Erwin Tam， Raghu Ramakrishnan， and Russell Sears. "Benchmarking cloud serving systems with YCSB." In Proceedings of the 1st ACM symposium on Cloud computing， pp. 143-154. 2010.

[5] Osemwegie， Omoruyi， Kennedy Okokpujie， Nsikan Nkordeh， Charles Ndujiuba， John Samuel， and Uzairue Stanley. "Performance Benchmarking of Key-Value Store NoSQL Databases." International Journal of Electrical and Computer Engineering 8， no. 6 (2018): 5333.

[6] 申德荣， 于戈， 王习特，等. 支持大数据管理的NoSQL系统研究综述[J]. 软件学报， 2013(08):96-113.

[7] 杨武军， 张继荣， 屈军锁. 内存数据库技术综述[J]. 西安邮电学院学报， 2005.

[8]Dimosthenis Kyriazis Cloud forward: From distributed to complete computing[J] Future Generation Computer Systems, 2018, 78

[9]Gregory Chockler, Seth Gilbert, Vincent Gramoli et al. Reconfigurable distributed storage for dynamic networks[J] Journal of Parallel and Distributed Computing, 2008, 69(1)

[10]Hamza Ben-Ammar, Yassine Hadjadj-Aoul, Gerardo Rubino et al. On the performance analysis of distributed caching systems using a customizable Markov chain model[J] Journal of Network and Computer Applications, 2019

[11] VMware官方网站[OD/DL] <https://www.vmware.com/>

[12] Redis官方网站[OD/DL] https://Redis.io/

[13] YCSB官方网站[OD/DL] https://github.com/brianfrankcooper/YCSB/

# 附录A 负载特征性能测试

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | workloada | | workloadb | | workloadc | | workloadd | | workloade | |
| 操作数 | 总时间 | 吞吐量 | 总时间 | 吞吐量 | 总时间 | 吞吐量 | 总时间 | 吞吐量 | 总时间 | 吞吐量 |
| 1000 | 171 | 5854 | 219 | 4571 | 164 | 6098 | 306 | 3268 | 3206 | 312 |
| 2000 | 252 | 7940 | 295 | 6783 | 328 | 6098 | 495 | 4040 | 6031 | 332 |
| 3000 | 454 | 6610 | 384 | 7815 | 443 | 6772 | 635 | 4724 | 10076 | 298 |
| 4000 | 509 | 7861 | 385 | 10392 | 383 | 10444 | 755 | 5298 | 15029 | 266 |
| 5000 | 447 | 11188 | 597 | 8377 | 480 | 10417 | 755 | 6623 | 18313 | 273 |
| 6000 | 601 | 9985 | 564 | 10640 | 662 | 9063 | 935 | 6417 | 22095 | 272 |
| 7000 | 780 | 8976 | 709 | 9874 | 677 | 10340 | 1047 | 6686 | 25899 | 270 |
| 8000 | 683 | 11714 | 780 | 10258 | 772 | 10363 | 1088 | 7353 | 29593 | 270 |
| 9000 | 833 | 10806 | 891 | 10102 | 805 | 11180 | 1218 | 7389 | 33702 | 267 |
| 10000 | 975 | 10257 | 968 | 10332 | 942 | 10616 | 1405 | 7117 | 36946 | 271 |
| 11000 | 953 | 11544 | 939 | 11716 | 991 | 11100 | 1263 | 8709 | 39978 | 275 |
| 12000 | 1107 | 10841 | 937 | 12808 | 936 | 12821 | 1553 | 7727 | 45811 | 262 |
| 13000 | 1047 | 12417 | 1172 | 11093 | 1169 | 11121 | 1458 | 8916 | 46369 | 280 |
| 14000 | 1080 | 12964 | 1146 | 12217 | 1153 | 12142 | 1206 | 11609 | 53442 | 262 |
| 15000 | 1193 | 12574 | 1209 | 12408 | 1358 | 11046 | 1377 | 10893 | 57121 | 263 |
| 16000 | 1327 | 12058 | 1373 | 11654 | 1399 | 11437 | 1419 | 11276 | 61669 | 259 |
| 17000 | 1355 | 12547 | 1452 | 11709 | 1460 | 11644 | 1608 | 10572 | 63961 | 266 |
| 18000 | 1495 | 12041 | 1565 | 11502 | 1477 | 12187 | 1524 | 11811 | 64195 | 280 |
| 19000 | 1549 | 12267 | 1520 | 12501 | 1522 | 12484 | 1778 | 10686 | 72177 | 263 |
| 20000 | 1457 | 13728 | 1508 | 13263 | 1624 | 12315 | 1809 | 11056 | 75897 | 264 |
| 21000 | 1513 | 13880 | 1649 | 12736 | 1770 | 11864 | 1813 | 11583 | 80030 | 262 |
| 22000 | 1723 | 12769 | 1751 | 12565 | 1580 | 13924 | 1938 | 11352 | 81558 | 270 |
| 23000 | 1614 | 14251 | 1838 | 12514 | 1952 | 11783 | 2058 | 11176 | 82666 | 278 |
| 24000 | 1687 | 14227 | 1994 | 12037 | 2003 | 11982 | 2052 | 11696 | 88253 | 272 |
| 25000 | 1851 | 13507 | 2057 | 12154 | 1992 | 12550 | 2086 | 11985 | 91607 | 273 |
| 26000 | 1897 | 13706 | 2116 | 12288 | 2186 | 11894 | 2757 | 9431 | 91037 | 286 |
| 27000 | 1973 | 13685 | 2187 | 12346 | 2224 | 12140 | 2491 | 10839 | 99046 | 273 |
| 28000 | 1923 | 14561 | 2280 | 12281 | 2270 | 12335 | 2272 | 12324 | 102150 | 274 |
| 29000 | 2080 | 13943 | 2405 | 12059 | 2369 | 12241 | 2496 | 11619 | 101797 | 285 |
| 30000 | 2086 | 14382 | 2620 | 11451 | 2402 | 12490 | 2828 | 10608 | 109723 | 273 |
| 31000 | 2179 | 14227 | 2640 | 11743 | 2466 | 12571 | 2838 | 10923 | 113587 | 273 |
| 32000 | 2254 | 14197 | 2656 | 12049 | 2535 | 12623 | 2687 | 11909 | 111724 | 286 |
| 33000 | 2337 | 14121 | 2817 | 11715 | 2607 | 12658 | 2799 | 11790 | 120451 | 274 |
| 34000 | 2464 | 13799 | 2794 | 12169 | 2821 | 12052 | 2880 | 11806 | 123963 | 274 |
| 35000 | 2475 | 14142 | 2851 | 12277 | 2722 | 12858 | 3024 | 11574 | 124296 | 282 |
| 36000 | 2396 | 15025 | 3009 | 11964 | 3023 | 11909 | 3113 | 11564 | 130871 | 275 |
| 37000 | 2580 | 14341 | 3067 | 12064 | 2938 | 12594 | 3169 | 11676 | 129711 | 285 |
| 38000 | 2619 | 14510 | 3128 | 12149 | 3098 | 12266 | 3031 | 12537 | 136950 | 277 |
| 39000 | 2688 | 14509 | 3058 | 12754 | 3146 | 12397 | 3296 | 11833 | 137967 | 283 |
| 40000 | 2848 | 14045 | 3123 | 12809 | 3243 | 12334 | 3238 | 12353 | 145444 | 275 |
| 41000 | 2993 | 13699 | 3335 | 12294 | 3172 | 12926 | 3458 | 11857 | 146246 | 280 |
| 42000 | 2906 | 14453 | 3449 | 12178 | 3382 | 12419 | 3646 | 11519 | 151582 | 277 |
| 43000 | 3074 | 13989 | 3529 | 12185 | 3423 | 12562 | 3577 | 12021 | 154380 | 279 |
| 44000 | 3142 | 14004 | 3748 | 11740 | 3506 | 12550 | 3701 | 11889 | 157106 | 280 |
| 45000 | 3216 | 13993 | 3664 | 12282 | 3446 | 13059 | 3842 | 11713 | 160204 | 281 |
| 46000 | 3268 | 14076 | 3713 | 12389 | 3656 | 12582 | 3867 | 11896 | 164912 | 279 |
| 47000 | 3502 | 13421 | 3767 | 12477 | 3636 | 12926 | 3879 | 12117 | 168342 | 279 |
| 48000 | 3562 | 13476 | 4000 | 12000 | 3833 | 12523 | 4031 | 11908 | 173823 | 276 |
| 49000 | 4001 | 12247 | 3983 | 12303 | 3821 | 12824 | 4130 | 11864 | 174569 | 281 |
| 50000 | 3663 | 13650 | 3953 | 12649 | 4044 | 12364 | 4215 | 11862 | 183699 | 272 |
| 51000 | 3630 | 14050 | 4203 | 12134 | 4037 | 12633 | 4273 | 11935 | 180505 | 283 |
| 52000 | 3918 | 13272 | 4215 | 12337 | 4126 | 12603 | 4424 | 11754 | 189039 | 275 |
| 53000 | 3778 | 14029 | 4398 | 12051 | 4271 | 12409 | 4431 | 11961 | 188117 | 282 |
| 54000 | 4098 | 13177 | 4309 | 12532 | 4224 | 12784 | 4515 | 11960 | 189975 | 284 |
| 55000 | 3821 | 14394 | 4579 | 12012 | 4371 | 12583 | 4361 | 12612 | 199087 | 276 |
| 56000 | 4108 | 13632 | 4552 | 12303 | 4469 | 12531 | 4490 | 12472 | 197201 | 284 |
| 57000 | 3778 | 15088 | 4746 | 12010 | 4504 | 12655 | 4311 | 13222 | 202592 | 281 |
| 58000 | 4417 | 13131 | 4417 | 13131 | 4691 | 12364 | 4846 | 11969 | 210481 | 276 |
| 59000 | 4419 | 13352 | 4734 | 12463 | 4690 | 12580 | 4983 | 11840 | 210316 | 281 |
| 60000 | 4363 | 13752 | 4727 | 12693 | 5477 | 10955 | 5008 | 11981 | 214087 | 280 |
| 61000 | 4507 | 13535 | 4931 | 12371 | 4866 | 12536 | 5000 | 12200 | 220541 | 277 |
| 62000 | 4763 | 13017 | 5383 | 11518 | 4856 | 12768 | 5028 | 12331 | 220433 | 281 |
| 63000 | 4560 | 13816 | 5135 | 12269 | 5087 | 12385 | 5258 | 11982 | 223187 | 282 |
| 64000 | 4752 | 13468 | 5176 | 12365 | 5243 | 12207 | 5364 | 11931 | 232211 | 276 |
| 65000 | 4743 | 13705 | 5312 | 12237 | 5270 | 12334 | 5390 | 12059 | 243276 | 267 |
| 66000 | 4942 | 13355 | 5291 | 12474 | 5120 | 12891 | 5461 | 12086 | 282523 | 234 |
| 67000 | 4921 | 13615 | 5538 | 12098 | 5277 | 12697 | 5597 | 11971 | 238243 | 281 |
| 68000 | 5280 | 12879 | 5346 | 12720 | 5318 | 12787 | 5563 | 12224 | 243322 | 279 |
| 69000 | 5227 | 13201 | 5694 | 12118 | 5522 | 12495 | 5257 | 13125 | 247935 | 278 |
| 70000 | 5545 | 12624 | 5645 | 12401 | 5510 | 12704 | 5115 | 13685 | 252096 | 278 |
| 71000 | 5359 | 13249 | 5450 | 13028 | 5609 | 12658 | 5170 | 13733 | 252592 | 281 |
| 72000 | 5331 | 13506 | 5819 | 12373 | 5568 | 12931 | 6078 | 11846 | 256667 | 281 |
| 73000 | 5379 | 13571 | 5875 | 12426 | 5690 | 12830 | 5891 | 12392 | 261352 | 279 |
| 74000 | 5766 | 12834 | 5934 | 12471 | 5749 | 12872 | 6101 | 12129 | 262881 | 281 |
| 75000 | 5467 | 13719 | 6168 | 12160 | 6047 | 12403 | 5840 | 12842 | 269509 | 278 |
| 76000 | 5303 | 14332 | 6111 | 12437 | 5779 | 13151 | 6394 | 11886 | 269244 | 282 |
| 77000 | 5764 | 13359 | 6483 | 11877 | 6194 | 12431 | 6323 | 12178 | 273685 | 281 |
| 78000 | 5178 | 15064 | 6405 | 12178 | 6130 | 12724 | 6412 | 12165 | 276865 | 282 |
| 79000 | 5754 | 13730 | 6504 | 12147 | 6294 | 12552 | 6468 | 12214 | 280377 | 282 |
| 80000 | 5971 | 13398 | 6619 | 12087 | 6363 | 12573 | 6624 | 12077 | 285822 | 280 |
| 81000 | 6033 | 13426 | 6520 | 12423 | 6497 | 12467 | 5689 | 14238 | 289360 | 280 |
| 82000 | 5831 | 14063 | 6619 | 12389 | 6384 | 12845 | 5784 | 14177 | 291764 | 281 |
| 83000 | 6215 | 13355 | 6700 | 12388 | 6560 | 12652 | 7062 | 11753 | 293702 | 283 |
| 84000 | 6066 | 13848 | 6793 | 12366 | 7331 | 11458 | 6868 | 12231 | 299174 | 281 |
| 85000 | 6258 | 13583 | 6660 | 12763 | 6828 | 12449 | 6953 | 12225 | 301012 | 282 |
| 86000 | 6460 | 13313 | 6761 | 12720 | 7377 | 11658 | 6897 | 12469 | 306638 | 280 |
| 87000 | 6516 | 13352 | 7080 | 12288 | 6889 | 12629 | 6992 | 12443 | 311688 | 279 |
| 88000 | 6521 | 13495 | 7043 | 12495 | 6863 | 12822 | 7108 | 12380 | 312887 | 281 |
| 89000 | 6734 | 13217 | 7183 | 12391 | 6852 | 12989 | 7110 | 12518 | 314743 | 283 |
| 90000 | 6739 | 13355 | 7185 | 12526 | 7067 | 12735 | 7292 | 12342 | 320678 | 281 |
| 91000 | 6733 | 13516 | 7168 | 12695 | 7246 | 12559 | 7280 | 12500 | 324646 | 280 |
| 92000 | 6767 | 13596 | 7728 | 11905 | 7126 | 12910 | 6036 | 15242 | 326423 | 282 |
| 93000 | 6392 | 14550 | 7380 | 12602 | 7167 | 12976 | 6549 | 14201 | 332691 | 280 |
| 94000 | 7082 | 13273 | 7667 | 12260 | 7332 | 12821 | 7571 | 12416 | 333417 | 282 |
| 95000 | 7067 | 13443 | 7675 | 12378 | 7342 | 12939 | 7812 | 12161 | 337170 | 282 |
| 96000 | 7175 | 13380 | 7821 | 12275 | 7490 | 12817 | 7705 | 12459 | 341046 | 281 |
| 97000 | 7024 | 13810 | 7900 | 12279 | 7741 | 12531 | 7798 | 12439 | 346335 | 280 |
| 98000 | 7137 | 13731 | 8158 | 12013 | 8174 | 11989 | 7944 | 12336 | 346027 | 283 |
| 99000 | 7153 | 13840 | 8068 | 12271 | 7819 | 12661 | 8530 | 11606 | 352781 | 281 |
| 100000 | 7107 | 14071 | 8189 | 12212 | 8166 | 12246 | 8395 | 11912 | 357034 | 280 |

# 附录B 测试脚本代码

# -\*- coding: utf-8 -\*-

import os

def rewrite\_workload(name, num\_turn):

# Use a breakpoint in the code line below to debug your script.

f = open("workloads/" + name, "w")

workloadstr = """

recordcount=%d

operationcount=%d

workload=com.yahoo.ycsb.workloads.CoreWorkload

readallfields=true

redis.host = 192.168.12.1

redis.port = 6379

readproportion=0.5

updateproportion=0.5

scanproportion=0

insertproportion=0

requestdistribution=zipfian

""" % (num\_turn, num\_turn)

f.write(workloadstr)

print('操作数修改为%d，修改完成' % (num\_turn))

f.close()

def load\_run(name):

os.system('bin/ycsb load redis -s -P workloads/%s' % (name))

os.system('bin/ycsb run redis -s -P workloads/%s>>output' % (name))

if \_\_name\_\_ == '\_\_main\_\_':

workload = "workloada"

num = 0

for i in range(10):

num += 1000

rewrite\_workload(workload, num)

load\_run(workload)

致谢

经过几个月的忙碌，我的毕业设计已经基本完成了。作为一名大四生，由于个人经验匮乏，难免有许多地方考虑不周全。如果没有窦晖老师的督促指导，以及学长们的建议提醒，要完成这个设计是十分困难的。在此，衷心感谢他们的支持与帮助。

其次，感谢大学四年来所有的老师，是他们为我打下了夯实的专业基础知识。没有这些专业基础，我不可能完成这次毕业设计，本次实验对于我来说不仅仅是数据库测试实验，同时也是一扇新的计算机学科的大门，带我见识到了除了高级语言以外计算机专业的其它领域知识技能，感受到了由操作系统不同所带来的软件运行效率显著不同，以及不同平台下，程序适配也有显著的不同。

最后感谢学院我的母校——安徽大学，这个将近四年以来给予我知识和关怀的地方。