分类号_____ 学号 M202077054

学校代码 10487 密级_____

华中科技大学

硕士学位论文

（学术型□ 专业型 ☑）

基于负载预测的键值存储智能弹性调度系统的设计与实现

学位申请人：陈仲为

学 科 专 业：计算机技术

指 导 教 师：谭志虎 教授

答 辩 日 期：2022 年 5 月 23 日

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Master Degree in Engineering

Design and Implementation of Key-Value Storage Intelligent Elastic Scheduling System Based on Load Prediction

Candidate : CHEN Zhongwei

Major : Computer Technology

Supervisor : Prof. TAN Zhihu

Huazhong University of Science and Technology

Wuhan 430074, P. R. China

May, 2022

摘 要

越来越多的企业在云上部署应用程序，并使用键值存储系统来存储应用数据。而应用负载波动通常具备一定的规律性，会在少数特定时间段处于高负载。针对波动负载，现有研究通过动态扩充和缩减存储资源，能够以最佳的存储资源满足负载的需要。但常用的被动调度策略具有明显的滞后性，扩充存储资源后进行数据迁移产生的迁移负载会与前台负载竞争，从而导致键值存储系统性能下降。如果能够准确预测应用负载的变化情况，提前执行调度策略，将能有效降低数据迁移对键值存储系统性能的影响。

针对被动调度产生的迁移负载与前台负载竞争导致性能下降的问题，提出了一种基于负载预测的键值存储智能弹性调度系统（Intelligent Elastic Scheduling System，IESS），该系统采用多种循环神经网络对负载数据进行建模，能准确预测云服务器中键值存储系统的负载变化，并基于预测负载信息，生成调度指令指导键值存储系统对存储资源的弹性调度。IESS 由智能调度模块、基础功能模块和智能预测模块组成，其中智能调度模块部署在云服务器的键值存储系统中，负责收集负载数据及执行调度指令；基础功能模块和智能预测模块部署在主机中，前者负责服务器配置、负载图像显示和服务器通信等基础功能，后者利用神经网络模型来预测未来一段时间的负载变化情况及生成调度指令。IESS 采用线下离线训练和线上在线训练结合方式，既能够尽可能减少对系统运行时的开销影响，又能够提高负载预测的准确度。

在互联网应用负载的场景下，针对 IESS 的负载预测效果和弹性调度效果进行详细测试。测试表明，在预测方面，DRNN_LSTM 模型的预测效果最好，对于出行服务应用和点餐服务应用两种数据集，误差百分比与决定系数（R2_Score）分别达到了 5.09%、98.29%和 7.20%、97.74%；在调度方面，可以在几乎不影响键值存储系统性能的情况下以较少的存储资源满足上层应用的运行需要，相比于无调度策略，IESS 弹性调度策略在两种负载场景下可以节省 33.27%和 52.13%的成本开销。

关键词：键值存储系统；神经网络模型；负载预测；弹性调度

Abstract

More and more enterprises are deploying applications on the cloud and using key-value storage systems to store application data. The fluctuation of application load usually has a certain regularity, and it will be under high load in a few specific time periods. For fluctuating loads, existing research can dynamically expand and reduce storage resources to meet the needs of the load with the best storage resources, but the commonly used passive scheduling strategy has obvious hysteresis, and the migration load generated by data migration after expanding storage resources will compete with the foreground load, which will cause the decline of performance of the key-value storage system. If the changes in application loads can be accurately predicted and the scheduling strategy implemented in advance, the impact of data migration on the performance of key-value storage systems can be effectively reduced.

In view of the problem of performance degradation caused by the competition between the migration load generated from passive scheduling and the foreground load, a key-value storage intelligent elastic scheduling system (IESS) based on load prediction is proposed. The system uses a variety of recurrent neural networks to model the load data, which can accurately predict the load changes of the key-value storage system in the cloud server, and generate scheduling instructions based on the predicted load information to guide the key-value storage system to elastically schedule storage resources. IESS consists of an intelligent scheduling module, a basic function module and an intelligent prediction module. The intelligent scheduling module is deployed in the key-value storage system of the cloud server and is responsible for collecting load data and executing scheduling instructions; the basic function module and intelligent prediction module are deployed in the host computer, the former is responsible for basic functions such as server configuration, load image display and server communication, and the latter uses neural network models to predict load changes in the future and generate scheduling instructions. IESS adopts a combination of offline training and online training, which can not only reduce the cost impact on system operation, but also improve the accuracy of load prediction.

In the scenario of the Internet application load, the load prediction effect and elastic scheduling effect of IESS are tested in detail. The test shows that in terms of prediction, the

DRNN_LSTM model has the best prediction effect. For the two data sets of travel service application and ordering service application, the error percentage and determination coefficient (R2_Score) reach 5.09%, 98.29% and 7.20%, 97.74% respectively; in terms of scheduling, it can meet the running needs of upper-layer applications with less storage resources without affecting the performance of the key-value storage system. Compared with no-scheduling strategy, the IESS elastic scheduling strategy can save 33.27% and 52.13% cost in two load scenarios.

Key words: Key-Value Storage System, Neural Network Model, Load Prediction, Elastic Scheduling

目　录

1 绪论

1.1 研究背景与意义

1.1.1 互联网应用快速发展

随着信息技术的高速发展，全球已经进入了数字经济时代，新一轮科技革命正在进行，互联网产业成为了国家重点扶持对象。为了加快数字经济的发展，国家持续不断地改进网络设施，鼓励科技创新，在互联网领域形成了一股创业浪潮，基础应用类、商务交易类、网络娱乐类以及公共服务类等互联网应用正在蓬勃发展，生态体系日益完善。根据中国信通院在 2021 年 11 月发布的《移动互联网应用程序（APP）个人信息保护治理》白皮书[1]显示，截至 2021 年 9 月底，应用程序在线数量高达 274 万款，应用程序分发量达到 20164 亿次。

根据中国互联网络信息中心（CNNIC）在 2022 年 2 月发布的第 49 次《中国互联网络发展状况统计报告》[2]显示，在全国 10.32 亿网民中，即时通信、网络视频、短视频用户使用率分别为 97.5%、94.5%和 90.5%，用户规模分别达 10.07 亿、9.75 亿和 9.34 亿，海量的用户使用让应用程序产生和处理的数据规模也在不断扩大，企业需要进一步提高大数据的存储和管理水平，为用户提供更优质的服务。

1.1.2 互联网应用逐渐云化

随着万物互联和智能化概念的提出，海量应用的不断创新对数据中心的计算和存储提出了更高的要求。然而，传统存储技术在面对这些新兴的应用程序时表现不佳，存在四个方面的不足：传统架构紧耦合和单一协议导致可扩展性差；资源调度无法统一导致资源浪费，成本高；交互能力差；运维难度大等。

---

[1] 《移动互联网应用程序（APP）个人信息保护治理白皮书》，2019.
http://www.caict.ac.cn/kxyj/qwfb/bps/202111/t20211119_392904.htm

[2] 第 49 次《中国互联网发展状况统计报告》，2022.
http://www.cnnic.net.cn/hlwfzyj/hlwxzbg/hlwtjbg/202202/t20220225_71727.htm

随着云技术的出现，计算和存储解耦，云平台成为高扩展、高可靠、快速恢复、高性能、按需付费的在线模式[1][2]，既满足了各种应用程序的计算和存储需求，也为一些传统应用提供了更高效的存储解决方案。因此，互联网应用上云受到了众多企业的重视。此外，受新冠肺炎疫情影响，远程办公、在线教育、疫情流调、疫情防控等云服务应用出现了飞速发展，同时也让全球数字化脚步加快，各主要国家均大力推进应用程序上云，国内外很多大型企业正在根据云平台的特点进行云上应用程序研发，例如 AWS 的 Amazon Aurora Serverless V2[3]，华为的云原生数据湖产品 FusionInsight MRS[4]，阿里巴巴的云原生数据库 PolarDB[5]，腾讯的云原生数据库 TDSQL-C[6]，形成了良好的带头示范作用，同时这些基础应用程序的出现加快了其它应用程序上云的脚步。

1.1.3 键值存储技术成为互联网应用重要的存储技术

被广泛应用的非关系型（NoSQL）数据库[7]可以高效地存储和管理海量数据，也是基于云平台的互联网应用的最优的数据存储引擎。键值存储技术是一种典型的 NoSQL 技术，在大数据中能够高效的进行数据索引操作，并且在海量存储系统中具有数据形式灵活、高可靠性、高度可扩展性、低延迟、低一致性开销和高吞吐量等特点，适合支持多种应用程序，已经成为了互联网应用重要的存储技术。

近年来，许多应用程序中写操作的百分比迅速增加[8]，Jinhong Li 等人通过对从阿里云收集到的数十亿个 I/O 请求的块级 I/O 数据进行分析[9]，同时对比十几年前的微软研究院发布的数据集后，发现云环境中大多数的应用程序中具有较多的写操作，而且普遍都是小 I/O 的随机请求。基于 LSM-tree[10]的键值存储技术能够将随机写聚合成顺序写，同时通过 Compaction 机制持续不断地在后台进行数据的合并和排序来保证数据有序，具有良好的读写性能，已经被互联网应用广泛采用，例如 BigTable[11]，HBase[12]，Cassandra[13]，CockroachDB[14]和 TiDB[15]。

1.1.4 互联网应用负载具有规律波动性

为了给用户提供优质的服务，同时降低生产成本，越来越多的企业在云上部署应用程序，并使用键值存储系统来存储应用数据。键值存储系统中可以存储不同应用程

序的数据，虽然它们的工作负载波动不同，但是从更长的时间观测来说，不同应用所产生的负载在整个数据库中往往存在一定规律的波动性。

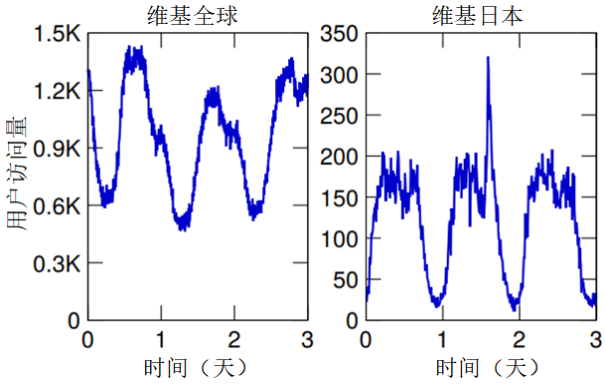

图 1.1　维基百科工作负载跟踪图[17]

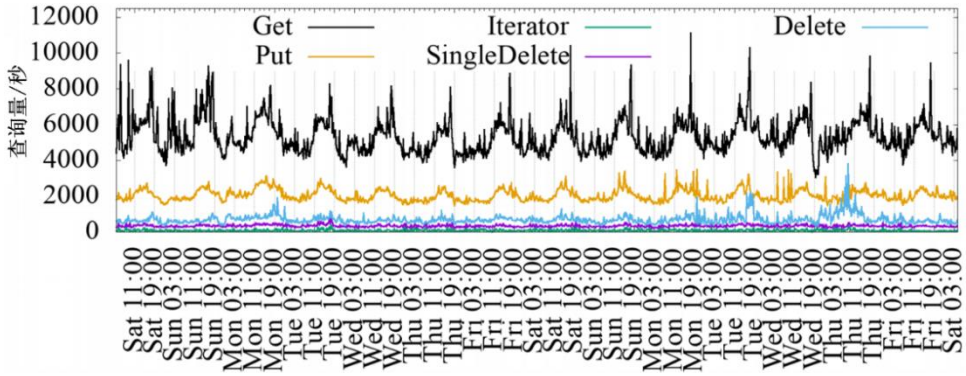

图 1.2　Facebook 各类查询的总体 QPS 变化图[18]

图 1.1 是维基百科工作负载跟踪图，图 1.2 是社交应用 Facebook 各类查询的总体 QPS 变化图，可以发现，两款应用每天的负载变化呈现出一定的规律性。

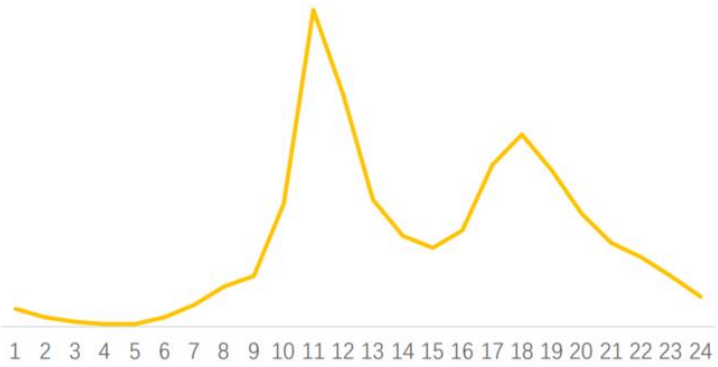

图 1.3　美团外卖订单下单时段分布图[1]

[1] 中国外卖产业调查研究报告（2019 年前三季度），2019. https://mri.meituan.com/research/report

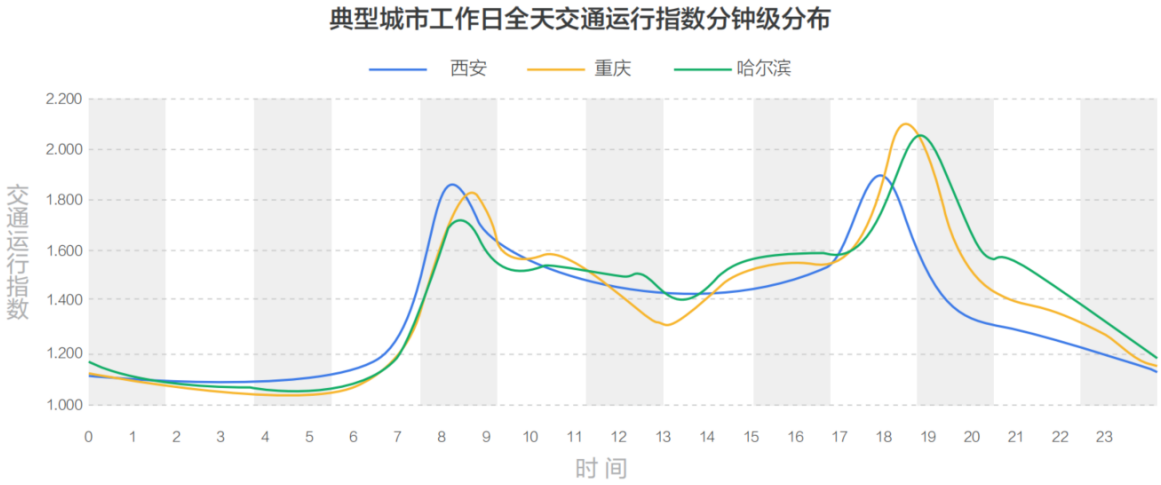

图 1.4　滴滴在几个城市工作日全天交通运行指数图[1]

图 1.3 是美团外卖订单下单时段分布图，图 1.4 是滴滴在在几个城市工作日全天交通运行指数图，图中的曲线表示服务负载量随时间的变化趋势，可以发现，两款应用在一天中的各个时间段的负载变化也呈现出一定的规律性。

从以上四个例子不难看出，这些互联网应用程序的负载高峰和负载低谷与用户的行为习惯息息相关，其工作负载具有明显的规律波动性。

云上互联网应用程序需要根据应用需求动态调整向云服务供应商购买的存储资源，当负载高峰即将来临时，从云平台购买更多的存储资源，以保证高的吞吐量和低时延，当负载即将进入低谷期时，逐渐减少存储节点，从而减少资源浪费并降低生产成本。互联网应用程序的工作负载决定着存储的需求，Mahendra P. Yadav 等人研究发现[16]，根据观察到的工作负载数据反馈式调整资源存在时间滞后的情况。因为应用程序向云平台申请资源需要一定的反应时间，因此提前预测工作负载更加重要。

针对被动调度策略中根据应用负载反馈式调整资源存在时间滞后的问题，本文提出一种基于负载预测的键值存储智能弹性调度系统，该系统利用神经网络模型对键值存储系统的负载进行准确预测，并基于预测负载信息，生成调度指令指导键值存储系统对存储资源的弹性调度。

[1] 城市交通出行报告（2019 第一季度），2019. https://max.book118.com/html/2019/1217/8105127136002070.shtm

1.2 国内外研究现状

1.2.1 负载预测的研究现状

国内外学者主要是基于传统模型和神经网络模型进行负载预测，下面将分别介绍基于这两类模型在负载预测方面的研究现状。

表 1.1 基于两类模型在负载预测方面的研究现状

模型类别	具体算法	研究目的
传统模型	稀疏回归分析模型[19]	预测未来时间段内的服务器负载情况
	梯度树提升（GTB）[20]	预测建筑物的冷却负荷
	MPIF-GBDT[21]	预测短期功率负载
	VRSP-LP[22]	基于负载预测来解决云计算环境下虚拟资源调度问题
	OMICFI-RFVS[23]	预测电力系统中的短期负荷
神经网络模型	CNN 模型[24]	预测通过数据中心网络流量的短期变化
	LSTM 模型[25]	预测云端游戏的工作负载来实现高效的资源分配
	iTLBO-ANN 模型[26]	预测城市级的电力负荷
	CSCWOA 算法[27]	预测短期云计算资源负载
	CGAN 模型[28]	对短期负荷进行预测

Takaya Miyazawa 等人利用一种稀疏回归分析模型[19]来对服务器的负载进行建模：先预设时间序列数据的稀疏性，然后使用最小绝对收缩和选择算子（LASSO）的建模方法来简化数学模型并通过减轻过度拟合来减少预测误差。经过多个迭代学习周期之后，预测模型可以有效地预测未来时间段内的服务器负载情况。

Haohan Sha 等人使用梯度树提升（GTB）[20]算法来预测建筑物的冷却负荷，并结合机械通风系统，利用较低的能源消耗来冷却高层建筑物。实验表明，由该算法生成的预测模型能够很好地预测建筑物的冷却负荷，在夏季可以减少机械通风系统 16%的能耗。

Shengwei Lv 等人使用基于梯度提升决策树的多特征池重要性融合（MPIF-GBDT）算法[21]来预测短期功率负载。MPIF-GBDT 由多特征池重要性融合和梯度提升决策树组成，前者提取不同特征选择方法的重要性值，并通过构造重要性向量、重要性矩阵

和权重分配重新组合它们，后者将得到的质量特征发送给预测引擎以提高预测精度。实验结果表明，MPIF-GBDT 算法可以准确预测短期功率负载。

王萍等人提出了 VRSP-LP[22]算法来解决云计算环境下虚拟资源调度问题。将最近一个时段内的虚拟机负载数据输入到 ARIMA 模型中，根据模型的预测结果来动态分配资源，以达到负载均衡的目的。实验结果表明，在虚拟机处于高负载状态的前提下，该算法既能有效降低虚拟机 CPU 或内存的占用率和虚拟机的性能损失。

郑睿程等人提出了 OMICFI-RFVS[23]算法来对电力系统的短期负荷进行预测。该算法使用 OMICFI 方法过滤高维数据集中的无关、噪声等特征，使用 RFVS 方法进一步优化所选择的特征值。实验表明，该算法能够有效提高短期负荷预测的精度。

Alberto Mozo 等人使用卷积神经网络（CNN）[24]来预测通过数据中心网络流量的短期变化。实验表明，对于数据中心网络的短期流量变化，基于神经网络的预测方法比传统的时间序列分析方法（如 ARIMA）表现更好，前者具有更高的准确度。

Koné K. Désiré 等人使用长短期记忆神经网络（LSTM）[25]技术来预测云端游戏的工作负载，在预测的基础上，使用基于骑手优化的和谐搜索算法来实现高效的资源分配。实验表明，该方法对云端游戏的多个性能指标显示出增强的效果。

Kangji Li 等人提出了一种新的混合预测模型——iTLBO-ANN 模型[26]，用于城市级电力负荷预测。利用 iTLBO 在全局范围内优化人工神经网络的参数以提高常规人工神经网络模型的准确性和鲁棒性。在数据预处理阶段，使用 DWT 用于消除原始数据的噪声和奇异值，使用 PCA 来细化最相关的输入变量。案例研究表明，iTLBO-ANN 模型在预测准确性方面表现出良好的性能。

谢建群等人在多层感知神经网络的基础上提出 CSCWOA 算法[27]，用来预测短期云计算资源负载。实验表明，CSCWOA 算法能够学习负载序列数据在各频段冲击毛刺的变化规律，在保持良好泛化能力的同时，还具有不错的预测精度。

林珊等人提出一种基于条件生成对抗网络（CGAN）模型[28]来对短期负荷进行预测。该模型中使用 CNN 来构建生成模型和判别模型，以负荷影响因素为条件来对网络进行博弈训练。实验表明，CGAN 模型可以兼顾一定的泛化能力并有效提高短期负荷的预测精度。

以上国内外学者在负载预测方面的研究表明，传统模型擅长处理线性数据，泛化能力较差，而神经网络模型具有良好的泛化能力，能够适应多种负载变化场景，尤其是神经网络模型中的 RNN 模型，擅长处理时序数据，对具有非线性特征的负载数据有着更高的预测精度。

1.2.2 键值存储技术的研究现状

键值存储（key-value store，KVS）由于数据管理简单、性能高，在工业界和学术界都得到了广泛的应用，成为了各类系统中必不可缺的基础存储组件。键值存储系统中最常用的索引结构是 B+tree 和 LSM-tree，下面将分别介绍国内外学者基于这两种索引结构在键值存储技术方面的研究现状。

表 1.2 基于两种索引结构在键值存储技术方面的研究现状

索引结构	系统	主要方法
B+tree	MongoDB[29]	使用基于数据局部性的在线高效流映射来扩展基于边界的流映射
	MongoDB[30]	针对大数据查询提出了一种分页查询技术的改进方法
	NLOV[31]	提供面向对象的块级存储接口，使用 NBD 和 AOE 作为传输协议
	MongoDB[32]	提出了一种基于节点实时负载的负载均衡改进算法
LSM-tree	LevelDB[33]	提出一种将键和值分开存储的结构
	RangeKV[34]	使用 NVM 中的 RangeTab 来管理 L0 数据并增加 L0 容量
	TRIAN[35]	减少维护 LSM-tree 结构时产生的写放大来提高 LSM KV 吞吐量
	SILK[36]	引入了 I/O 调度控制器，提高客户端操作反应速度
	PebblesDB[37]	FLSM 引入守卫的概念来组织日志，避免在同一级别重写数据

Trong D. Nguyen 等人提出一种基于 MongoDB 独特特性的新型流映射方案[29]，来解决当前市场上流行的文件存储 MongoDB 中存在的问题。使用基于数据局部性的在线高效流映射来扩展基于边界的流映射，将每种文件类型映射到不同的流中，来解决集合文件和索引文件具有不同生命周期以及存在数据碎片的问题。测试结果表明，该方案可以将 YCSB 和 Linkbench 的吞吐量分别提高 44%和 43.73%。

Jian Hu 等人针对 MongoDB 在处理大数据时跳转数据的分页显示效率太低的问题，提出了一种分页查询技术的改进方法[30]。该方法在原有的 skip-limit 技术中加入

“条件查询+排序+限时返回记录”方法，以提高分页查询的速率。测试结果表明，改进的分页查询方法大大提高了 MongoDB 对大数据的查询速率。

Lu Qi 等人在 BerkeleyDB 的基础上提出了一种面向对象的存储系统 NLOV[31]。NLOV 提供了一个块级的面向对象的存储接口，使用 NBD 和 AOE 作为传输协议，并支持两种协议之间的交互操作。测试结果表明，NLOV 具有良好的简洁性、更好的灵活性和适用性。

陈敬静等人为了解决 MongoDB 中节点的负载均衡问题，提出了一种基于节点实时负载的负载均衡改进算法[32]。该算法在节点上增加负载代理来监测每个节点的负载情况，并将每个节点的负载指数作为 chunk 块迁移的一个指标。实验表明，该算法能够有效均衡各个节点的负载，显著提高集群的并发读写性能。

Lei Wang 等人在基于 LevelDB 的基础上，提出一种将键和值分开存储的结构[33]，使用值的偏移量和 LevelDB 中存储的长度信息将值存储在一个单独的文件中。测试结果表明，优化后 LevelDB 的顺序写入性能降低了约 40%，随机写入和覆盖性能提高了 200%以上。

Ling Zhan 等人在基于 RocksDB 的基础上，使用 NVM 来优化 KV 存储并提出 RangeKV[34]。RangeKV 使用 NVM 中的 RangeTab 来管理 L0 数据并增加 L0 容量，来减少 LSM-tree 级别和系统压缩的数量。测试结果表明，与 RocksDB 相比，整体随机写入吞吐量提高了 4.5 到 5.7 倍。

Diego Didona 等人提出了一种基于 LSM-tree 的新的 KV 存储——TRIAN[35]。TRIAN 通过减少维护 LSM-tree 结构时产生的写放大来提高 LSM KV 吞吐量。在 LSM 内存组件级别，TRIAN 利用数据流行度的偏差来避免对最流行的键进行频繁的 I/O 操作；在存储级别，TRIAN 通过延迟和批处理多个 I/O 操作来均摊管理成本；在日志提交级别，TRIAN 避免了对存储的重复写入。测试结果表明，TRIAN 可将吞吐量提高多大 193%，可将写放大减小多达 4 倍，并将 I/O 减少了一个数量级。

Oana Balmau 等人提出了 SILK[36]，该系统在 RocksDB 的基础上，引入了 I/O 调度控制器。SILK 通过动态的 I/O 调节，在高负载期间给客户端更多带宽，在低负载期间限制内部操作和面向客户操作之间的资源竞争，提高了客户端操作反应速度。测

试结果表明，SILK 的第 99 百分位延迟比 RocksDB 和 TRIAN 低两个数量级。

Pandian Raju 等人受跳表这种数据结构的启发，提出了一种全新的数据结构：分段日志结构合并树（FLSM）。他们使用 FLSM 来对 HyperLevelDB 进行二次开发，构建了一个新的键值存储系统：PebblesDB[37]。FLSM 使用守卫的概念来组织日志，尽量避免在同一层（level）内对数据进行重写。测试结果表明，与 RocksDB 相比，PebblesDB 将写放大减小了 2.4-3 倍，同时将写吞吐量提高了 6.7 倍。

以上国内外学者在键值存储技术方面的研究表明，采用 B+tree 作为索引结构对读性能更加友好，而采用 LSM-tree 作为索引结构对写性能更加友好。由于云环境中大多数应用程序具有较多的写操作，因而以 LSM-tree 作为索引结构的键值存储系统被互联网企业广泛使用，其相关研究逐渐成为主流。

1.2.3 弹性调度技术的研究现状

弹性调度技术能够根据云环境中应用程序的工作负载情况，动态调整所分配的存储资源，使其在满足用户需求的前提下，尽可能降低总体成本。因此，该技术受到众多企业的重视，下面将介绍国内外学者在弹性调度技术方面的研究现状。

Philipp Hoenisch 等人提出了一种可以动态、弹性地调整虚拟机和容器配置的控制架构[38]。在该架构下，虚拟机和容器可以根据目标函数的搜索来动态更改实例的数量或调整用于实例的计算资源，来实现最小化成本的目的。测试实验表明，该控制架构可以降低 20%至 28%的成本。

Chuanqi Kan 设计了一种基于 Docker 的 Web 应用弹性云平台——DoCloud[39]。他使用 ARMA 模型来预测 Web 应用程序的工作负载，并结合 CPU 和内存使用的静态阈值，动态调整分配给 Web 应用程序的 Docker 容器数量。实验表明，DoCloud 具有良好的效率和可扩展性。

Chenggang Wu 等人设计了一种能够在云上自动伸缩、分层的分布式键值存储系统——Anna[40]。他们通过增加和移除节点来调整服务规模，通过将冷数据降级来降低成本，并且以细粒度来处理扩展请求。评估表明，Anna 的效率极高，相比于流行的云存储服务系统更具有成本效益。

以上国内外学者在弹性调度技术方面的研究表明，在云环境下，弹性调度技术具

有广阔的应用场景，能够大大降低应用程序所需的存储成本。

1.3 存在的问题

互联网应用负载的波动性给资源分配带来了严重的挑战，运维人员需要根据观察和经验判断进行资源的调度分配，存在效率低等问题。若能根据负载变化情况为应用动态分配存储资源，将为企业节省大量成本开销。

结合研究背景分析，并综合当前国内外研究现状，目前负载预测研究和云上的键值存储技术研究主要存在以下几点问题：

（1）互联网应用负载波动具有一定的规律性，根据工作负载数据反馈式调整资源存在调度滞后的情况，调度期间迁移负载与前台负载产生竞争，导致性能下降。

（2）神经网络模型在负载预测方面要优于传统模型，但在相同应用负载环境下，缺乏针对不同的循环神经网络模型之间的对比分析。

（3）采用 LSM-tree 作为索引结构的键值存储系统更适合云上数据存储，但无法根据负载变化情况来动态分配存储资源。

1.4 相关技术介绍

1.4.1 负载预测模型介绍

循环神经网络擅长处理时序数据，因此，IESS 中采用多种循环神经网络来实现负载预测模型。循环神经网络（Recurrent Neural Network，RNN）所有的循环单元按链式结构进行连接，并且能以时序数据作为输入，在时间序列的演进方向进行递归计算[41]。RNN 的研究开始于 20 世纪八十至九十年代，并在 21 世纪初发展为深度学习领域的重要算法之一[42]。相比于其它神经网络，RNN 具有记忆性，更擅长学习时序数据中的非线性特征。

（1）长短期记忆神经网络

长期依赖问题普遍存在于一般 RNN 中，严重影响模型的预测精度。为此，人们设计出长短期记忆（Long Short Term Memory，LSTM）神经网络来解决这个问题。研究 LSTM[43]的论文最早发表于 1997 年，相比于 RNN，LSTM 通过 cell 的状态来记

忆信息，采用三种特殊的门结构并结合激活函数来降低梯度消失和梯度爆炸的可能性，能够同时处理短期依赖和长期依赖这两类问题。

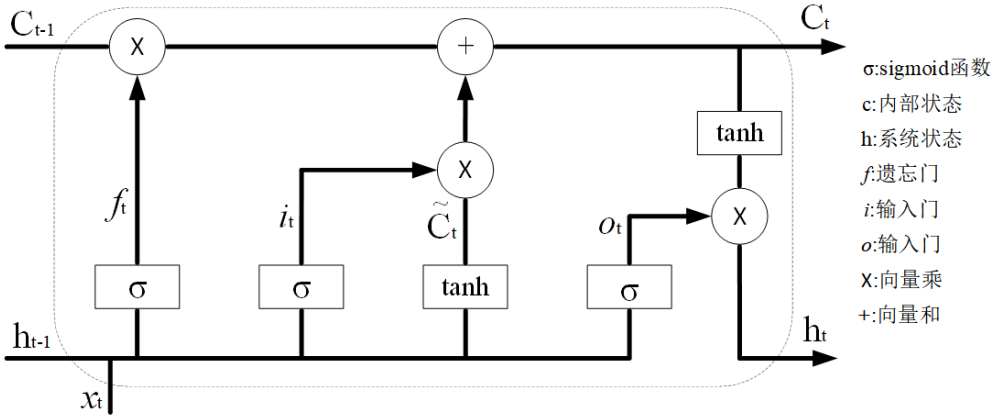

图 1.5　LSTM 隐藏层结构

LSTM 的隐藏层结构比较复杂，如图 1.5 所示。内部核心由遗忘门、输入门和输出门这三个“门”结构组成，在它们的相互作用下，当前时刻的c_t和h_t由前一时刻的c_{t-1}、h_{t-1}和当前时刻的输入x_t所决定。

（2）门控循环单元神经网络

与 LSTM 的设计初衷相同，门控循环单元（Gated Recurrent Unit，GRU）神经网络也是为了解决长期依赖、梯度消失和梯度爆炸等问题。研究 GRU[44]的论文最早发表于 2014 年，相比于 LSTM，GRU 删去了对提升学习能力贡献最小的输出门，精简了神经网络结构，从而大大提高对时序数据的学习效率。

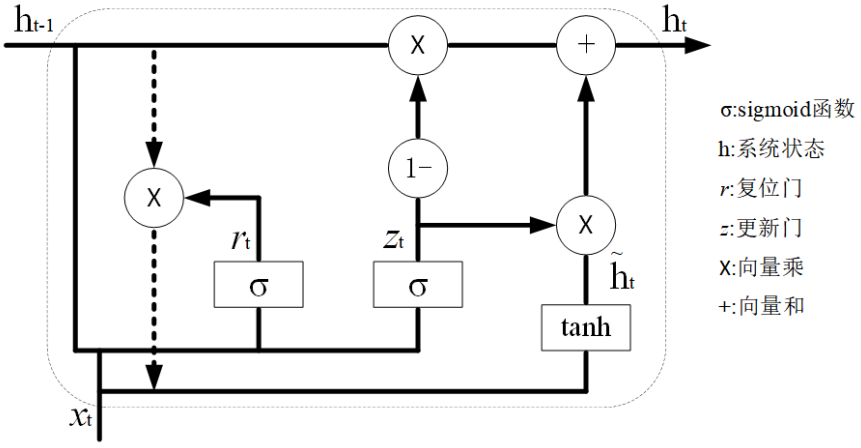

图 1.6　GRU 隐藏层结构

不同于 LSTM 的 3 个门， GRU 的隐藏层结构中只有复位门和更新门，如图 1.6 所示。其中复位门与 LSTM 中输入门的功能相似，负责控制前一时刻状态信息的输入，更新门则结合了输出门和遗忘门的功能，负责控制前一时刻状态信息对当前时刻状态信息的影响程度，即更新记忆。

（3）深度循环神经网络

标准的 RNN 只有一层隐藏层，网络结构简单但是性能较差，将多个 RNN 单元堆叠起来就能构成深度循环神经网络[45]（Deep Recurrent Neural Network，DRNN）。复杂的神经网络结构使得隐藏层能够拟合更加复杂的非线性函数，从而大大提高预测模型的准确度。参与构建 DRNN 的 RNN 单元可以是标准 RNN、LSTM 和 GRU 等。使用简单循环网络构建的 DRNN 也被称为循环多层感知器[46]。

（4）双向循环神经网络

由于一般的 RNN 都是从前往后来处理时序数据，在当前时刻无法注意到未来的上下文信息，进而无法适应现实生活中的某些应用场景，双向循环神经网络[47]（Bidirectional Recurrent Neural Network，BRNN）就是为了解决这个问题而提出来的。

BRNN 的隐藏层由向前推算层和向后推算层组成，向前推算层与单向的 RNN 相似，向后推算层则与标准的 RNN 按照时间序列反向传播相似。和 DRNN 类似，BRNN 也可以由各种类型的 RNN 构成，由 LSTM 构成的版本被称为双向 LSTM[48]。

1.4.2 RocksDB 系统介绍

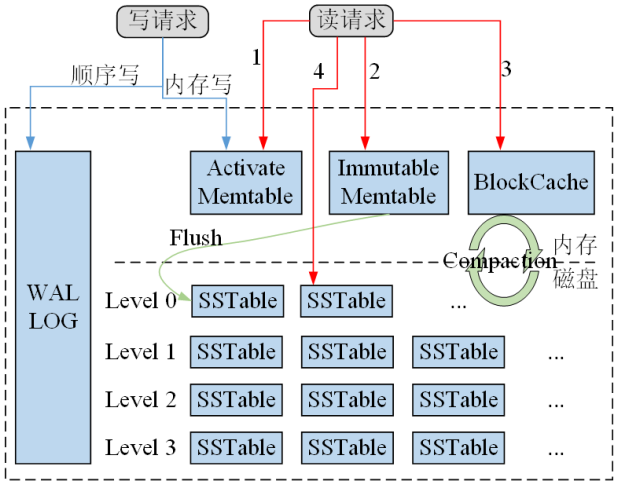

图 1.7 RocksDB 基本结构图

RocksDB 的写效率非常高，可以满足云上数据存储的需求，因此，IESS 中采用 RocksDB 作为存放数据的键值存储系统。RocksDB 是基于 LevelDB 研发的高性能键值持久化存储引擎，以库组件形式嵌入程序中，为大规模分布式应用在 SSD 上运行提供优化。RocksDB 使用日志结构合并树（LSM-tree）作为基本的数据存储结构，RocksDB 的基本结构如图 1.7 所示。

当写请求进入 RocksDB 后，首先会写到磁盘上的 Write-Ahead-Log（WAL）中进行持久化，然后将数据写入到内存中的 MemTable 缓存中，若 MemTable 缓存的数据量达到预设值，MemTable 与 WAL 将会转为不可变状态，同时分配新的 MemTable 与 WAL 用于后续写入，接着将不可变 MemTable 中相同的 Key 进行合并，生成 SSTable 数据文件后 flush 到磁盘上，并丢弃关联的 MemTable 与 WAL 数据。

LSM-tree 有多个层级，每个层级由多个 SSTable 组成，最新的 SSTable 会放置在 L0 层，下层的 SSTable 通过 Compaction 操作创建。每层 SSTable 总大小由配置参数决定，当 L 层数据大小超出预设值，会将 L 层的 SSTable 与 L+1 层的 SSTable 重叠部分进行合并，通过这一过程将写入数据从 L0 逐渐推到最后一层。L0 中单个 SSTable 的 Key 是有序的，不同的 SSTable 存在 Key 重叠，其余层的数据是整体有序的。

当读请求进入 RocksDB 后，首先从多个 MemTable 开始搜索，接着从 SSTable 数据文件的 L0 层依次往下层搜索，直至查到指定的 Key。在 LSM-tree 中，除 L0 层需要遍历每个 SSTable 外，其余层使用二分查找来搜索目标 SSTable，接着利用布隆过滤器（Bloom Filter）过滤掉非必要搜索的 SSTable，最后在 SSTable 中使用二分查找来搜索指定 Key 的 Value。

1.5 本文主要内容

信息技术的高速发展带动了互联网应用的快速发展，越来越多的企业在云上部署应用，并使用键值存储系统来存储应用数据。用户的行为习惯决定了互联网应用负载具有规律波动性，即在少数特定时间段处于高负载，而在其余时间段处于低负载。针对被动调度产生的迁移负载与前台负载竞争导致的性能下降问题，本文提出一种基于负载预测的键值存储智能弹性调度系统（IESS），来实现云服务器中键值存储系

统对存储资源的弹性调度。该系统主要由智能预测模块、基础功能模块和智能调度模块组成，后续章节将对这些模块的设计和实现做详细的阐述。

本文的主要内容包括以下五个章节：

第一章首先介绍了论文的研究背景与意义，接着介绍了国内外学者对于负载预测和键值存储技术的研究现状，然后指出现有研究中存在的问题，之后介绍了实现IESS 所用到的关键技术，最后阐述了本文的主要内容。

第二章首先分析真实互联网应用负载具有规律波动性，然后分析被动调度策略存在的问题，针对这个问题，给出具体的解决方案，最后详细介绍 IESS 的设计方案及设计细节。

第三章详细介绍 IESS 每个功能模块的实现过程，主要以流程图、表格等方式进行阐述。

第四章阐述整个系统的测试过程，针对负载预测模型和键值存储系统智能弹性调度进行详细测试，通过测试结果，得出相应结论。

第五章对本文的工作进行概括性的总结，分析研究过程中存在的不足之处，并对未来的研究方向进行展望分析。

2 智能弹性调度系统设计

本章首先针对真实互联网应用负载进行分析，阐述负载具有规律波动性，进而分析键值存储系统现有调度策略存在的缺陷。针对被动调度产生的迁移负载与前台负载竞争导致的性能下降问题，提出一种基于负载预测的键值存储智能弹性调度系统（Intelligent Elastic Scheduling System，IESS），IESS 能够预测负载变化趋势并进行资源调度。最后详细介绍该系统的设计思路及设计细节。

2.1 问题分析

2.1.1 真实互联网应用负载具有规律波动性

在现有的各种商业数据库中，由于用户的行为习惯往往具有规律性，反映到数据库层面就是各种应用的数据请求也往往呈现一定的规律性。参考《中国外卖产业调查研究报告》[1]和《城市交通出行报告》[2]，一类典型的互联网应用负载请求变化曲线如图 2.1 所示：

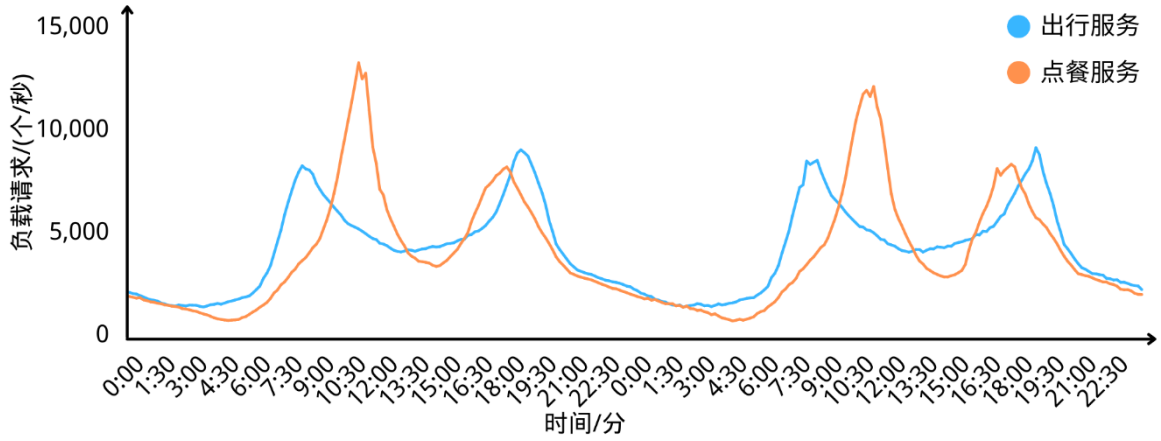

图 2.1 互联网应用负载请求变化趋势

图 2.1 表示的是连续两个工作日里，出行服务应用和点餐服务应用在不同时间段里的负载请求变化趋势，横纵坐标分别表示时间和负载请求大小，蓝色曲线表示的是

[1] 中国外卖产业调查研究报告（2019 年前三季度），2019. https://mri.meituan.com/research/report

[2] 城市交通出行报告（2019 第一季度），2019. https://max.book118.com/html/2019/1217/8105127136002070.shtm

出行服务应用，橙色曲线表示的是点餐服务应用。可以发现这两种互联网应用在相邻两天内的负载请求变化趋势十分相似，具有明显的规律波动性。同时，这两种互联网应用有着各自的负载高峰与负载低谷：出行服务应用的负载高峰出现在早上7:00~9:00 和晚上的 18:00~19:00，这正好对应着人们上班与下班的时间；点餐服务应用的负载高峰出现在中午 11:00~12:00 和晚上的 17:30~18:30，且中午的峰值明显高于晚上的峰值，这说明人们更倾向于回家吃晚餐。

相对于低负载来说，高负载只会出现在特定时间段，持续的时间也明显更短。一天当中，高负载开始的时间并不是确定的，也会存在些许波动，但整体上是规律的。

通过上述分析可以得知，在实际生活中，商业数据库中的负载请求变化具有规律波动性，不同应用的负载请求变化有着各自不同的峰值和谷值时间，这与用户的行为习惯息息相关。

2.1.2 被动调度策略影响键值存储系统的性能

大多数互联网应用中具有较多的写操作[9]，在负载高峰来临时，存储资源的不足会使得大量操作产生竞争，导致应用的响应速度变慢，从而影响用户的操作体验。由于互联网应用的负载请求往往具有规律波动性，负载高峰只会出现在特定时间段内，在其余时间段仍然是低负载状态，若企业长期为该应用分配大量的存储资源，虽然该应用不会因负载高峰而出现延迟卡顿的现象，但这会极大增加企业的运营成本。

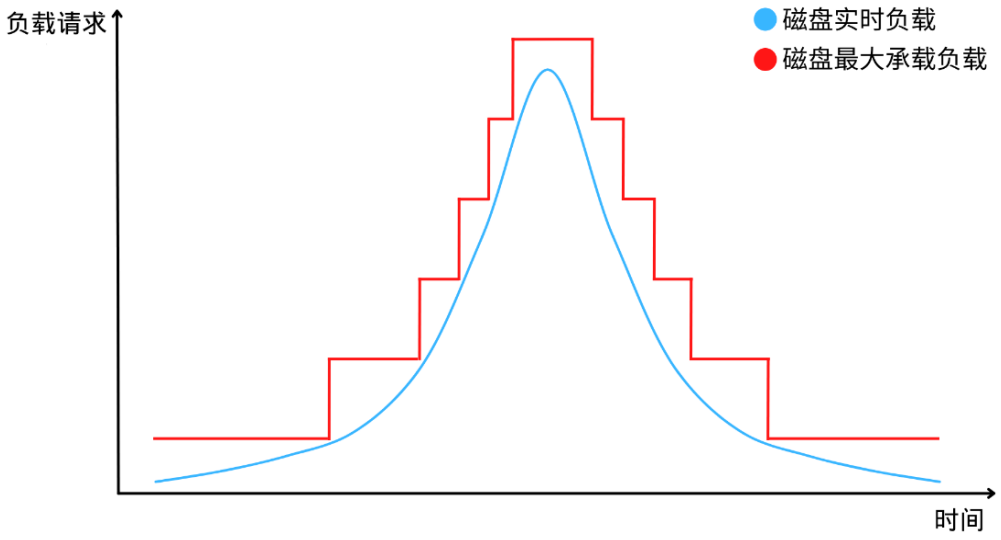

图 2.2 弹性调度示意图

一种比较好的解决方案是对云服务器中键值存储系统所使用的存储资源进行弹性调度，在负载请求上升时扩充存储资源，在负载请求下降时缩减存储资源，如图 2.2 所示。图中蓝色曲线表示负载请求，红色曲线表示存储资源，可以看到红色曲线是阶梯型上升和下降的，对应着存储资源的扩充和缩减。

但是存储资源弹性调度的关键是要把握好调度的时机，现有的调度策略大多数是被动调度，即只有当负载达到一定阈值时才会进行调度操作，具有明显的滞后性。数据迁移会产生大量的读写操作，在被动调度的情形下，这些读写负载会与键值存储系统中已经到来的前台负载高峰产生竞争，从而导致键值存储系统的性能下降。被动数据迁移示意图如图 2.3 所示，前台负载与迁移负载进行叠加后会产生更大的负载，影响键值存储系统的性能。

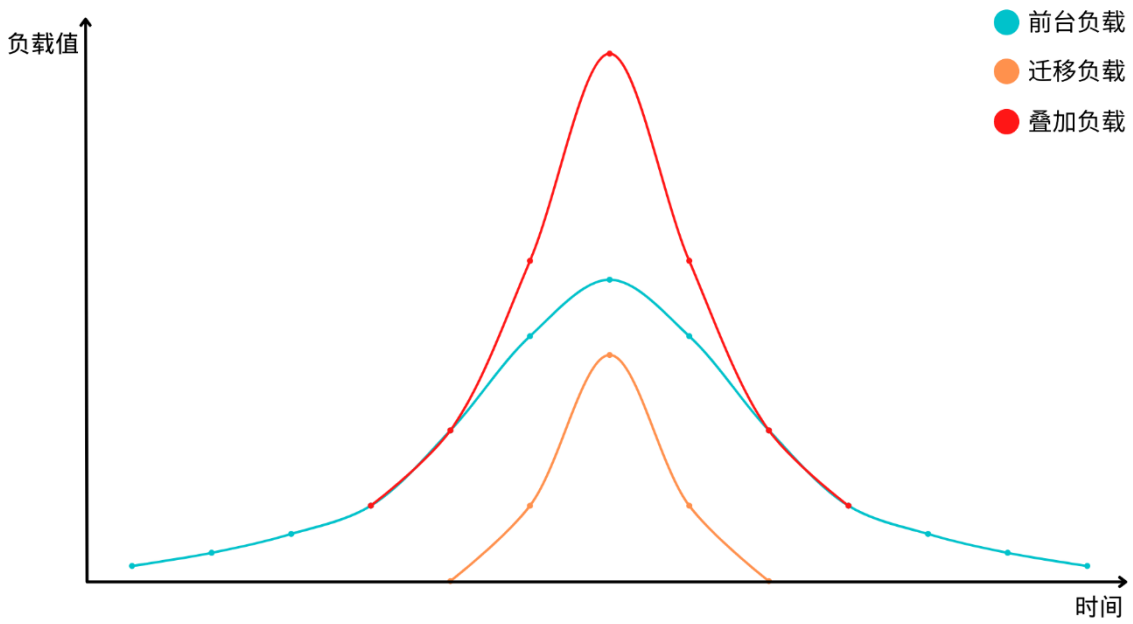

图 2.3　被动数据迁移示意图

从上述分析可知，存储资源的被动调度策略很难满足实际生活中的应用场景，因此需要设计一种弹性调度算法，在低负载时刻提前执行调度操作，避免数据迁移产生的读写负载与即将到来的前台负载高峰产生竞争，在负载请求下降时，对之前扩充的存储资源进行缩减，降低企业的运营成本。

2.2 智能弹性调度系统整体设计

2.2.1 设计思想

由问题分析可知，互联网应用的负载变化具有规律波动性，即在少数特定时间段处于高负载，而在其余时间段处于低负载。对于企业来说，为保证时长较短的负载高峰期内的请求服务质量，从而配置大量存储资源会带来很大的成本和资源浪费。一种比较好的方法是实现动态分配存储资源，但现有的调度策略大多数是被动调度，即只有当负载达到一定阈值时才会进行调度操作，具有明显的滞后性，扩充存储资源后进行数据迁移产生的迁移负载会与前台负载竞争而导致键值存储系统性能下降。如果能在应用的访问负载即将出现峰值时提前执行调度操作，这样就可以有效避免因数据迁移而导致键值存储系统性能下降的问题，在降低企业运营成本的同时保证服务质量。

负载数据本质上就是一种时序数据，具有很强的时间关联性。而 RNN 擅长处理时序数据，可以有效地提高预测准确度。因此本文的目的就是要借助 RNN 对时序数据的准确预测，设计并实现一种基于负载预测的键值存储智能弹性调度系统（IESS）。IESS 通过智能调度模块来获取云服务器中键值存储系统的实时负载请求，利用智能预测模块来预测未来一段时间内的负载请求变化情况。若预测出即将出现高负载请求，系统会给予相应的提示信息，并向智能调度模块发送调度指令来实现键值存储系统对存储资源的弹性调度。

IESS 会在线上环境中动态更新负载预测模型来保证负载预测的准确度，在模型更新的过程中，进行负载预测时仍会使用旧的预测模型，模型更新完毕后，新旧模型会自动替换，不影响系统的正常运行。IESS 根据调度指令扩充存储资源后，需要将相应的 SSTable 数据文件迁移到新扩充的存储资源中，在数据迁移过程中，键值存储系统仍然会访问现有存储资源中的 SSTable 数据文件，而不会访问新扩充存储资源中的数据，数据迁移完毕后，新旧路径信息自动替换，键值存储系统才会访问新扩充存储资源中的 SSTable 数据文件。

2.2.2 整体设计

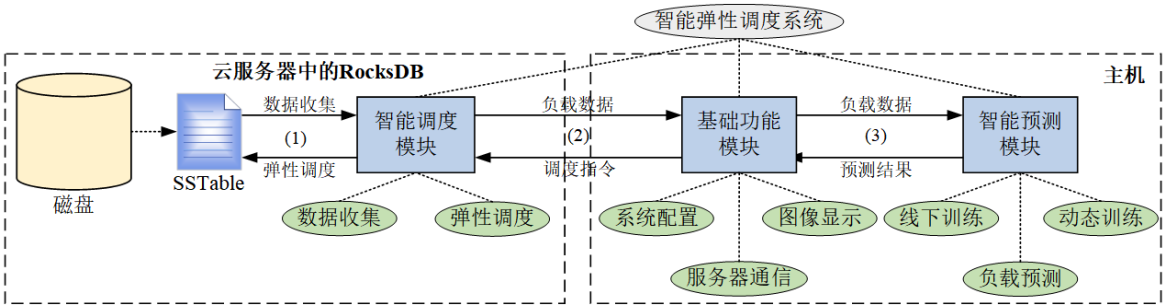

图 2.4 IESS 整体设计图

IESS 的整体设计如图 2.4 所示。该系统主要由三个模块组成：智能预测模块、基础功能模块和智能调度模块，其中智能调度模块部署在云服务器中的 RocksDB 内，基础功能模块和智能预测模块部署在主机中。

各个模块的功能如下：

（1）智能调度模块负载收集云服务器中 RocksDB 的实时负载请求，并根据系统发送的调度指令，将 SSTable 数据文件迁移到扩充的存储资源中，实现键值存储系统对存储资源的弹性调度；

（2）基础功能模块为系统的正常运行提供基础功能支持，包括对连入系统的服务器进行配置、负载曲线图像显示、向服务器请求负载数据和发送调度指令等；

（3）智能预测模块在线下环境利用数据集提前训练预测模型，在线上环境根据实时负载请求数据对预测模型进行动态训练以及预测未来一段时间内的负载请求变化情况。

2.2.3 智能弹性调度系统特点分析

IESS 采用多种循环神经网络对负载数据进行建模，能准确预测云服务器中键值存储系统的负载变化，并基于预测负载信息，生成调度指令指导键值存储系统对存储资源的弹性调度。IESS 与现有最新工作的不同之处在于：

（1）采用多种循环神经网络对负载数据进行建模，选择其中预测效果最好的 RNN 模型来构建 IESS 的智能预测模块，并在线上环境中动态训练，保证负载预测的准确度。

（2）基于负载预测信息来指导键值存储系统对存储资源的弹性调度，避免了被

动调度产生的迁移负载与前台负载竞争导致性能下降的问题。

2.3 智能预测模块设计

2.3.1 设计思想

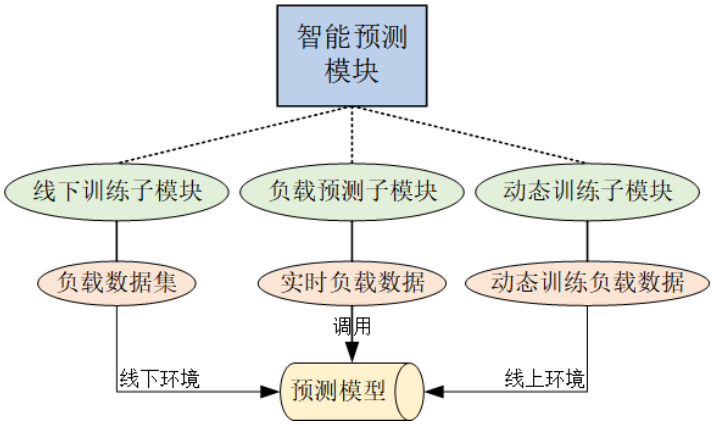

图 2.5 智能预测模块组成

智能预测模块主要实现与负载预测相关的功能，其组成如图 2.5 所示。该模块主要由三个子模块组成：线下训练子模块、负载预测子模块和动态训练子模块。其中线下训练子模块在线下环境使用负载数据集对预测模型进行训练，提高预测精度；负载预测子模块调用预测模型，根据实时负载数据对未来一段时间内的负载变化情况进行预测；动态训练子模块在线上环境使用动态训练负载数据对预测模型进行动态训练，提高预测精度。下面将详细介绍这些子模块是如何设计的。

2.3.2 线下训练子模块设计

为保证未进行动态训练的预测模型能够在线上环境中发挥一定的预测作用，需要提前在线下环境中使用典型的互联网应用数据集来训练神经网络模型。未经处理的原始数据集无法直接用于模型训练，因此需要先对数据集进行数据处理，然后再对已经构建好的神经网络模型进行训练。

（1）线下训练流程设计

预测模型线下训练流程设计如图 2.6 所示。本文采用机器学习中 RNN 的训练方法，通过重复迭代训练不断优化神经网络模型中的参数，最终得到预测精度较高的训练模型。

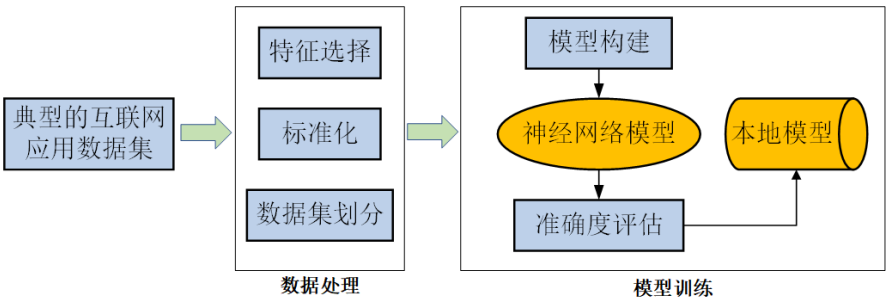

图 2.6　预测模型线下训练流程设计

预测模型线下训练过程主要分为以下 4 个步骤：

（1）对数据集进行数据处理，选取所需要的特征值，对特征值进行标准化处理后，按照一定比例划分成训练集和测试集；

（2）使用 TensorFlow 框架搭建多个 RNN 模型；

（3）设计模型训练方法，使用处理后的数据集来训练模型；

（4）对训练的模型进行预测准确度评估，将准确度较高的模型保存为本地模型。

下面主要从数据处理、模型训练和预测准确度评估这三个方面来介绍预测模型线下训练的流程。

（2）数据处理设计

本文采用读请求与写请求的和作为特征值，该特征值能够反映互联网应用的负载变化特点。考虑到预测的实际效果，负载数据需以分钟作为基本单位，即采用每分钟内读请求与写请求的和作为负载预测的特征值。

在模型训练前对数据进行标准化处理，能够避免个别数据项对模型带来较大偏差，本文采用 Z-Score 标准化方法[49]来对数据进行处理，通过式（2.1），经过处理的数据符合标准正态分布：

$$X_{normal} = \frac{x - \mu}{\sigma} \tag{2.1}$$

其中$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$，$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$，式（2.1）中的 x 是特征的原始值，μ 和 σ 分别是该特征在数据集中的均值和标准差。

（3）模型训练设计

训练过程主要分为三个步骤：a.定义神经网络结构，根据该网络结构计算预测值；

b.设计损失函数和反向传播优化算法；c.使用训练数据集反复训练。损失函数的值能够反映模型训练的效果，损失函数的值越小，则表示模型训练的效果越好。因此，模型训练的目的就是要不断减小损失函数的值。

在本文中，选择常用的交叉熵损失[50]作为损失函数，假设 m、n 是两个概率分布，m、n 交叉熵根据式（2.2）计算：

$$H(m,n) = -\sum_x m(x) \cdot log\, n(x) \qquad (2.2)$$

在逻辑回归问题中，m 可以表示真实结果的概率分布，而 n 可以表示逻辑回归结果的概率分布，H(m,n)表示 m 和 n 之间的交叉熵大小，反映了两个概率分布之间的相似程度，若该值越小，表示 m 和 n 越接近，逻辑回归的结果也就越准确。

反向传播优化算法有很多种，TensorFlow 框架中也有很多与之对应的优化器。本文采用的是常用且有效的 Adam 优化器，实践表明，Adam 整体上比其它优化方法的效果要好[51]。

在本文中，采用多维单步的时序数据预测方式来训练神经网络模型，训练过程中所涉及的主要参数有 time_step、batch_size 和 predict_step，其中 time_step 指的是时间跨度，batch_size 指的是每次训练的一个批次的大小，predict_step 指的是预测跨度。

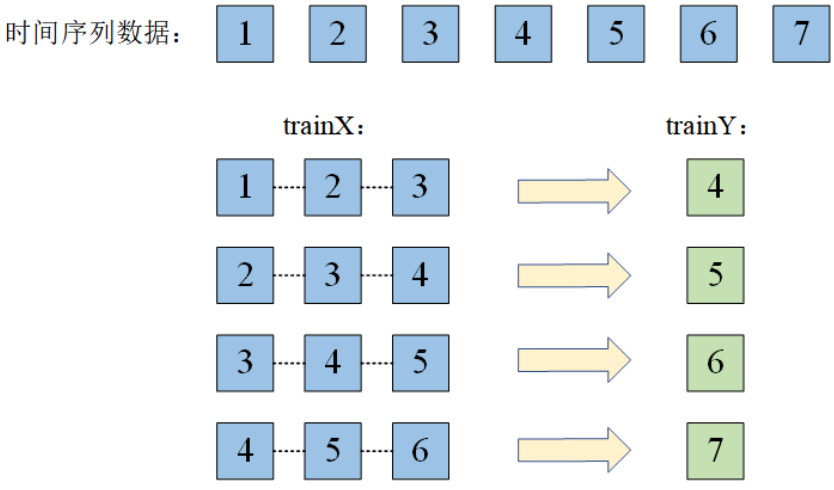

图 2.7　多维单步的时序数据预测

以图 2.7 为例，此时有 time_step=3，batch_size=4，predict_step=1，对于给定的时序数据，需要将 trainX 时序数据和 trainY 时序数据同时输入到神经网络模型中，模型会根据 trainX 时序数据输出一个值，并将该值与对应的 trainY 时序数据进行比

对，通过反向传播优化算法不断修改神经网络模型中的参数来缩小二者之间的差距，提高模型预测的准确度。

（4）预测准确度评估设计

本文采用统计学中判断回归模型有效性的三种常用方法：平均绝对误差（MAE）、平均平方根误差（RMSE）和决定系数（R2_Score），来评估神经网络模型对负载预测的准确度。若用y_i表示实际的观测值，$\bar{y}$表示实际观测值的均值，$\hat{y}_i$表示模型输出的预测值，则它们可以根据式（2.3）、（2.4）、（2.5）来计算：

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\hat{y}_i - y_i| \tag{2.3}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{2.4}$$

$$\mathrm{R2\_Score} = \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.5}$$

本文通过对比各个神经网络模型的预测负载曲线与真实负载曲线的拟合度以及计算它们之间的误差等方式来评估负载预测的准确度，并选择其中负载预测效果最好的神经网络模型来构建智能预测模块。

2.3.3 负载预测子模块设计

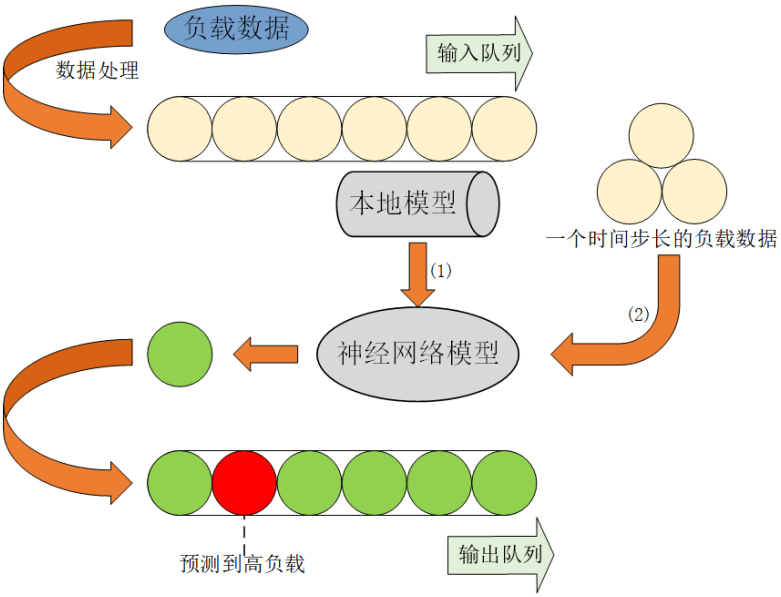

图 2.8　负载预测流程设计

该子模块能够根据输入队列中的负载请求数据来进行负载预测，负载预测过程

如图 2.8 所示，主要完成以下操作：

（1）读取模型：读取本地保存的模型文件，恢复上一次训练后的神经网络模型到内存中；

（2）负载预测：从输入队列中取出一个时间步长的负载数据输入到神经网络模型中进行预测，将得到的模型输出值放入到输出队列中，对于预测到的高负载数据，系统会给予相应提示信息并生成调度指令。

在本文中，相关参数设置为 time_step=20，predict_step=20，所以每次从输入队列中取出的一个时间步长的值为基于当前时间的前面 20 分钟的负载请求数据的值（20 组数据），将这些值放入到神经网络模型中，经过计算处理后得到 20 分钟后的负载预测值并放入到输出队列中。预测每分钟进行一次，每次预测完后都会从输入队列的队首删除 1 组数据。同时，该子模块会自动检查输出队列中的预测值，若预测到高负载数据，还会将该异常结果发送给基础功能模块，系统会给予相应提示信息并生成调度指令。

2.3.4 动态训练子模块设计

在线上真实环境中，线下训练所用数据集与真实环境不匹配或真实环境中的负载数据复杂多变，会导致已有预测模型的预测效果较差，因此需要在线上真实环境中对预测模型进行动态训练来提高预测的准确度。预测模型的动态训练也需要经过数据处理和模型训练两个过程。

在线上的真实环境中，该子模块会调用基础功能模块所存储的实时负载请求数据，动态训练过程如图 2.9 所示，主要完成以下操作：

（1）读取模型：读取本地保存的模型文件，恢复上一次训练后的神经网络模型到内存中；

（2）模型训练：当输入队列的长度达到指定阈值后，将输入队列拆分成训练集和测试集，并将训练集放入模型中动态训练，以提高预测的准确度；

（3）保存模型：当本次预测模型训练结束之后，保存训练好的模型文件。

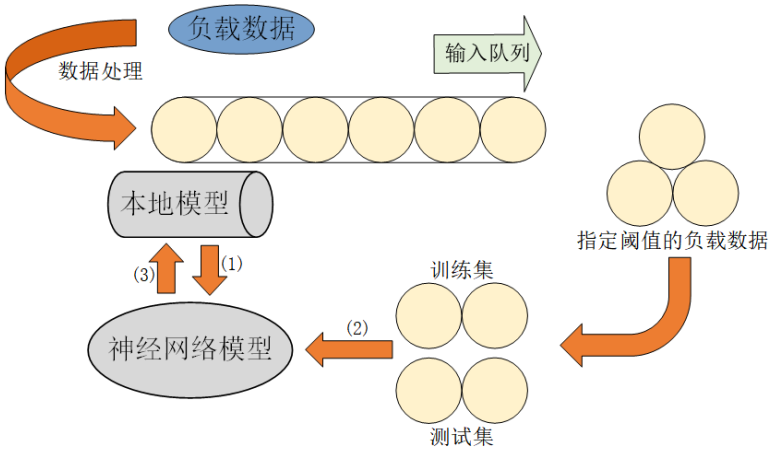

图 2.9　预测模型动态训练流程设计

该子模块会维护一个输入队列来存放云服务器中键值存储系统的实时负载请求数据，所有的负载请求数据在放入对应的输入队列前都需要进行数据处理。当输入队列的长度达到指定阈值后，将输入队列按照 9:1 的比例拆分成训练集和测试集来对神经网络模型进行动态训练。输入队列中存放的数据是以分钟为单位，此处输入队列的阈值设定为 300 分钟所接收的数据量，即按照 1 分钟 1 个数据，300 分钟便有 300 个数据。动态训练时的参数设置为：time_step=20，predict_step=20，batch_size=30，即根据前面 20 步来预测下一步，预测的是 20 分钟后的负载，每次训练的一个批次大小为 30。

2.4　基础功能模块设计

2.4.1　设计思想

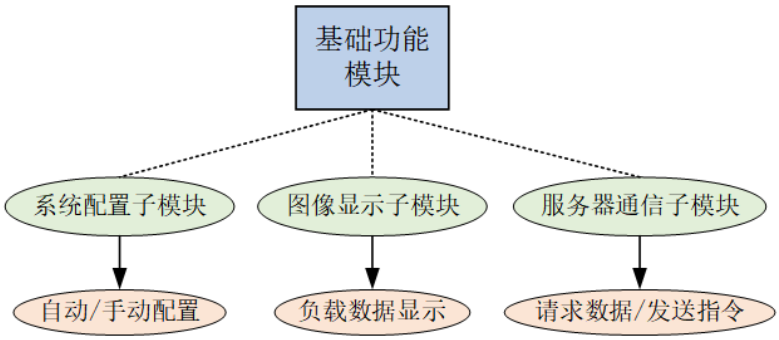

图 2.10　基础功能模块组成

基础功能模块负责为系统的正常运行提供基础功能支持，其组成如图 2.10 所示。

该模块主要由三个子模块组成：系统配置子模块、图像显示子模块和服务器通信子模块。其中系统配置子模块通过自动或手动方式配置接入系统的服务器信息，包括服务器名称和 IP 地址；图像显示子模块以图像的形式显示实时负载数据和预测负载数据；服务器通信子模块负责与服务器进行通信，即向服务器请求负载数据和发送调度指令，并将负载数据进行解析和存储，以供其它模块调用。下面将详细介绍这些子模块是如何设计的。

2.4.2 系统配置子模块设计

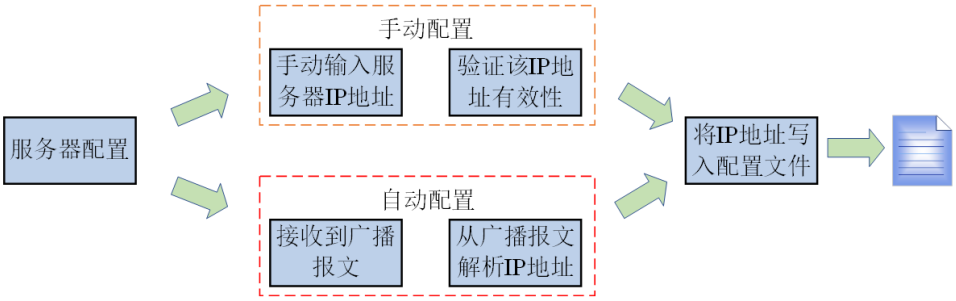

图 2.11 服务器配置流程设计

服务器的配置方式分为自动配置和手动配置，如图 2.11 所示。当服务器和部署系统的主机在同一个子网下时，系统会采用自动配置的方式：服务器中的智能调度模块会将本机的 IP 地址广播到网络中，系统接收到信息后会自动将该 IP 地址写入到配置文件。若不在同一个子网下时，系统只能采用手动配置的方式，手动输入的 IP 地址通过验证后会被写入到配置文件。

服务器的自动配置使用 UDP 协议中的广播功能来实现，部署在服务器上的智能调度模块会自动将本机的 IP 地址广播给同一子网上的所有主机，而系统在接收到该 IP 地址后会自动更新内存中保存的所有服务器信息，并将该信息持久化到配置文件。

服务器的手动配置则需提前知晓目标服务器的 IP 地址，手动输入到系统中，该 IP 地址通过有效性验证后会自动对内存中的服务器信息进行更新，然后再持久化到配置文件。

内存中所保存的服务器信息始终是最新的，因此可以直接对内存中保存的服务器信息进行查询、修改和删除操作，然后再将该信息持久化到配置文件中即可。

2.4.3 图像显示子模块设计

该子模块能够将服务器中键值存储系统的负载请求变化情况以图像的形式显示出来，负载请求以分钟作为基本单位。系统能够同时监控多台服务器的负载请求变化情况，通过下拉框的方式手动选择想要查看的服务器。在显示的负载请求变化图像中，实时负载曲线在前，预测负载曲线在后，两条曲线随着时间的推移而不断延伸，直至达到设定的最大长度。

2.4.4 服务器通信子模块设计

该子模块能够根据内存中保存的服务器信息与连入系统的服务器进行通信，即通过 TCP 协议向这些服务器请求负载数据和根据负载预测结果发送调度指令。由于网络状况存在不确定性，因此每次套接字连接请求需设置一个时间限制，若在该时间限制内未收到反馈则视为该 IP 地址不可达。该子模块采用线程的方式向服务器请求负载数据来提升系统的运行性能。

负载数据是通过 TCP 协议在网络上进行传输的，会涉及到数据安全性问题，在传输过程中必须进行加密处理，因此该子模块在接收后必须进行数据解析。除此之外，该子模块还承担着数据仓库的功能，存储着其它模块所需要调用的数据。

2.5 智能调度模块设计

2.5.1 设计思想

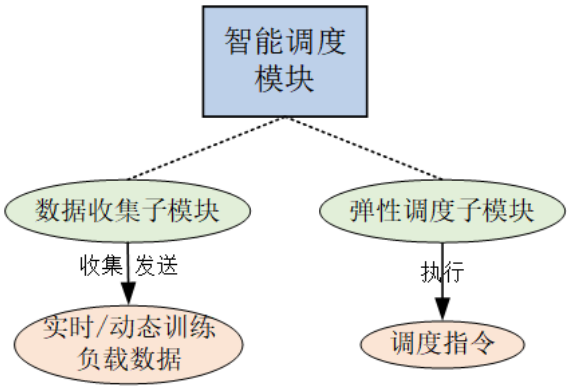

图 2.12 智能调度模块组成

智能调度模块负责收集实时负载请求数据及执行调度指令，其组成如图 2.12 所

示。该模块主要由两个子模块组成：数据收集子模块和弹性调度子模块。其中数据收集子模块通过拦截键值存储系统的读写请求来收集实时负载数据和动态训练负载数据，并根据数据请求的类型发送不同类型的负载请求数据；弹性调度子模块根据系统发送的调度指令来实现键值存储系统对存储资源的弹性调度。下面将详细介绍这些子模块是如何设计的。

2.5.2 数据收集子模块设计

智能调度模块需部署在云服务器中的键值存储系统中，通过拦截键值存储系统的读写请求来收集负载请求数据。预测模型动态训练所需的数据量较大，收集这些数据所需的时间较长，而部署系统的主机不可能花费这么长的时间去等待，因此用于动态训练的数据需要提前存储在云服务器中。所以该子模块除了收集该云服务器中键值存储系统的实时负载请求数据外，还需收集用于预测模型动态训练的负载数据。

该子模块在上层应用的读请求与写请求通过 RocksDB 访问磁盘内的 SSTable 数据文件前会进行拦截统计，统计的内容为每秒钟 RocksDB 的读写请求总和，具体过程如图 2.13 所示。

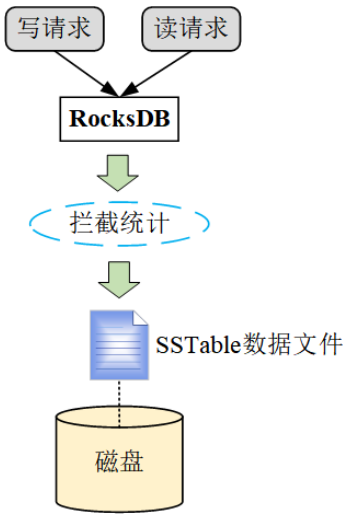

图 2.13　读写请求拦截示意图

该子模块使用两个数据队列来存放键值存储系统的实时负载请求数据，具体存放过程如图 2.14 所示。数据队列 1 中存放以秒为单位的实时负载请求数据，而数据队列 2 中存放以分钟为单位的实时负载请求数据。当数据队列 1 中的数据达到 60 个

时，将这 60 个数据求和取平均值，然后将该值插入数据队列 2 的队尾，并将数据队列 1 进行清空处理。数据队列 2 的最大长度设置为 1440，即 24 小时的数据，超过该长度时则会删除队首的数据（最旧的数据）。

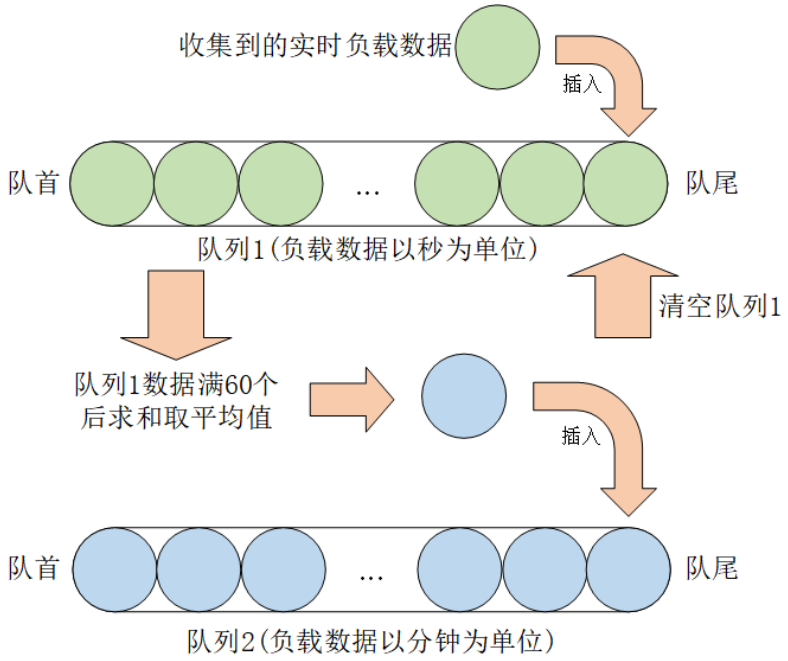

图 2.14　实时负载请求数据存放示意图

数据队列 2 中存放着实时负载数据与动态训练负载数据，如图 2.14 所示。其中，数据队列 2 的队尾数据是最新的，即为实时负载数据，数据队列 2 的所有数据为动态训练负载数据。当接收到实时负载数据的请求时，该子模块会取出最新插入队列的负载数据，即队尾的数据，将该数据按照一定的数据格式加密后通过 TCP 协议发送回系统；当接收到动态训练负载数据的请求时，该子模块会判断存放该数据的队列长度，若队列长度达到阈值，则将该数据连同实时负载数据一起加密后通过 TCP 协议发送回系统，否则只会发送实时负载数据。

2.5.3　弹性调度子模块设计

云服务器中键值存储系统所保存的数据以数据文件的形式保存在磁盘中，数据文件是弹性调度的基本单位，其中数据文件又可以分为热点数据和非热点数据。为应对即将出现的应用访问负载高峰，键值存储系统需提前进行调度操作，将热点数据迁移到新扩充的存储资源中。

热点数据会随着时间的推移而不断变化，因而需要保证数据热度的时效性。为此，本文设计了一个简单的热度随时间衰减的指数函数。若数据文件 i 被访问的总次数为 n，并且在最近一段时间Δt内被访问的次数为 m，初始时刻设置为T_0，则数据文件 i 在当前时刻 T 的热度值 H 可以根据式（2.6）计算：

$$H_i(T) = \frac{m+\frac{n-m}{4}}{e^{T-T_0}} \tag{2.6}$$

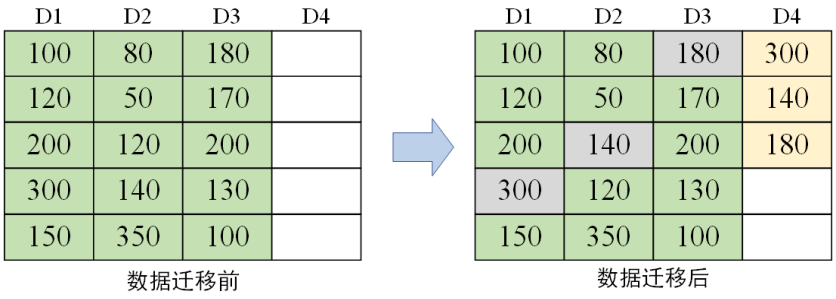

图 2.15　数据迁移示意图

在收到 IESS 的调度指令后，键值存储系统会进行调度操作，其中数据迁移的示意图如图 2.15 所示，图中的 D1 至 D4 表示键值存储系统中的 4 个磁盘，其中 D4 为新扩充的磁盘，绿色表示原始数据文件，灰色表示被迁移走的数据文件，黄色表示新迁移的数据文件，数据文件上的数字表示的是热度值。

以图 2.15 为例，扩充存储资源后，键值存储系统在进行数据迁移前会计算磁盘 D1 至 D3 中每个数据文件的热度值，其中磁盘 D1 至 D3 的热度值为 870、740 和 780，期望进行数据迁移之后可以达到热度值均衡的效果，即所有磁盘的热度值差异不大，因而通过计算可得四个磁盘的平均热度值为 597.5。根据该平均热度值可以知道磁盘 D1 需要迁移热度值为 300 的数据文件、磁盘 D2 需要迁移热度值为 140 的数据文件、磁盘 D3 需要迁移热度值为 180 的数据文件，这样磁盘 D1 至 D4 的热度值分别为 570、600、600 和 620，大致达到热度值均衡的目的。

当上层应用的负载请求较低时，键值存储系统会自动缩减存储资源而不依赖于 IESS 的调度指令。当要缩减存储资源时，键值存储系统会将之前迁出的数据文件迁回到原来的磁盘中。

2.6 本章小结

本章首先针对真实互联网应用负载进行分析，说明负载变化具有规律波动性，接着分析键值存储系统现有调度策略存在的缺陷。基于以上分析，提出了一种基于负载预测的键值存储智能弹性调度系统，利用神经网络模型对键值存储系统未来的负载变化情况进行准确预测，并根据预测结果生成调度指令指导键值存储系统进行资源配置，使得键值存储系统能够在前台负载高峰来临前完成存储资源扩充和数据迁移的任务，避免被动的调度策略而导致键值存储系统性能下降，同时在负载降低后收缩存储资源，降低生产成本。

3 智能弹性调度系统实现

本章主要介绍 IESS 是如何实现的。首先给出了 IESS 的整体架构图，然后分别介绍 IESS 的各个模块是如何实现的，详细地介绍其中涉及的数据结构、功能算法实现以及主要流程实现。

3.1 智能弹性调度系统整体架构

IESS 的整体架构如图 3.1 所示。该系统主要由三个模块组成：智能预测模块、基础功能模块和智能调度模块，其中智能调度模块部署在云服务器中的 RocksDB 内，负责数据收集和弹性调度的功能；基础功能模块和智能预测模块部署在主机中，前者负责系统配置、图像显示和服务器通信的功能，后者负责线下训练、负载预测和动态训练的功能。

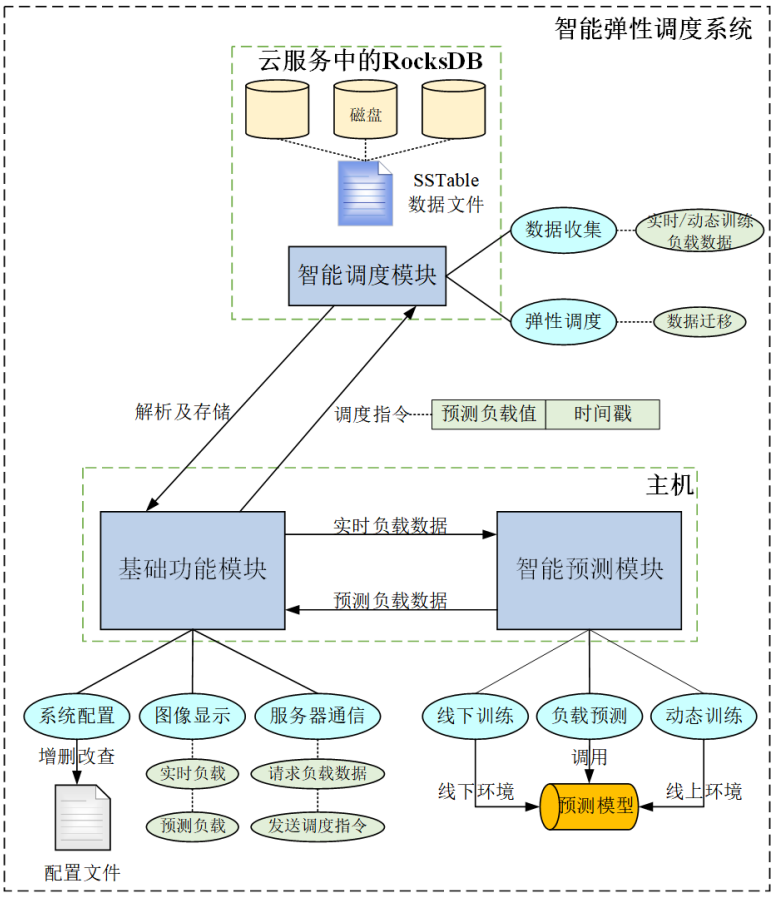

图 3.1 IESS 整体架构图

智能调度模块会对 RocksDB 的读写请求进行拦截统计，并将负载请求数据保存在数据队列中，根据不同的数据请求发送不同的负载数据。若收到系统发送的调度指令，该模块会根据公式计算出磁盘中每个 SSTable 数据文件的热度值，确定待迁移的 SSTable 数据文件，然后将待迁移的 SSTable 数据文件迁移到新扩充的存储资源中。

基础功能模块能够以自动配置和手动配置两种方式来配置连入系统的服务器信息，根据配置文件的信息能够向连入系统的服务器发送数据请求以及调度指令，对于接收到的负载数据，该模块会进行解析、图像显示并按照一定格式存储在内存中，以供其它模块调用。

智能预测模块能够调用基础功能模块中存储的负载数据，通过动态训练来提高神经网络模型的预测精度，并利用该模型对未来一段时间内的负载请求变化情况进行预测，将预测的结果返回给基础功能模块进行显示。若预测出即将出现高负载请求，系统会给予相应的提示信息，并由基础功能模块向智能调度模块发送调度指令来实现键值存储系统对存储资源的弹性调度。

下面将详细介绍智能预测模块、基础功能模块和智能调度模块是如何实现的。

3.2 智能预测模块实现

智能预测模块主要实现与负载预测相关的功能，该模块主要由三个子模块组成：线下训练子模块、负载预测子模块和动态训练子模块，下面将详细介绍这些子模块是如何实现的。

3.2.1 线下训练子模块实现

（1）数据处理

数据处理的具体处理过程如图 3.2 所示。在本文中，使用典型的互联网应用数据集来进行线下训练，数据集中的每条数据按照“应用单元标识/读写类型/逻辑地址/时间戳”的格式进行排列，例如“0,r,20941264,0.551706”这条数据表示的意思是应用单元标识为 0 在 0.551706 秒请求对逻辑地址 20941264 进行读操作。原始数据集不能直接用来进行预测模型的训练，需将这些数据处理为每分钟内读请求个数和写请求个数的总和，图 3.2 中的文件 2 即为处理后的数据集，可以直接用来训练预测模型。

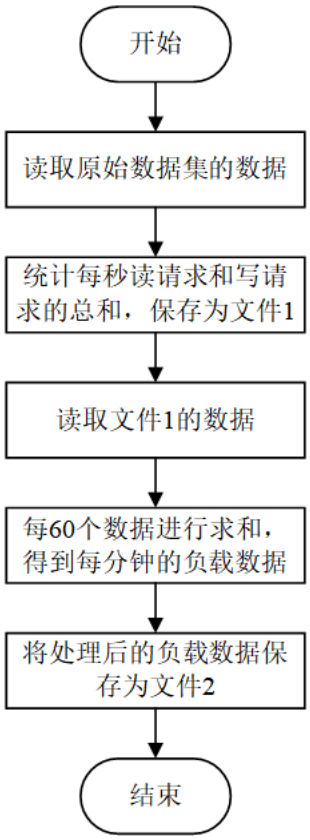

图 3.2　数据处理流程图

（2）神经网络构建

本文中所要构建的 RNN 有标准 RNN、LSTM、GRU、DRNN 和 BRNN，使用 TensorFlow 框架，能够比较轻松地构建出这些神经网络。通过对这些 RNN 模型进行对比测试，选择其中负载预测效果最好的 RNN 模型来构建智能预测模块。

使用 tf.nn.rnn_cell.BasicRNNCell 函数来构建一个标准 RNN 的神经网络单元，得到的神经网络单元需要再通过 tf.nn.dynamic_rnn 函数来封装成标准 RNN。tf.nn.rnn_cell.BasicRNNCell 函数和 tf.nn.dynamic_rnn 函数的参数及返回值如表 3.1 和表 3.2 所示。

表 3.1　tf.nn.rnn_cell.BasicRNNCell 参数及返回值

变量标识	变量作用	变量说明
num_units	函数参数	隐藏层节点数
cell	函数返回值	返回一个神经网络单元

表 3.2 tf.nn.dynamic_rnn 参数及返回值

变量标识	变量作用	变量说明
cell	函数参数	神经网络单元
inputs	函数参数	输入的训练数据
initial_state	函数参数	神经网络单元的初始状态
dtype	函数参数	数据类型
output_rnn	函数返回值	最后的输出结果
final_states	函数返回值	最后一个 cell 的状态

使用 tf.nn.rnn_cell.LSTMCell 函数来构建一个 LSTM 的神经网络单元，得到的神经网络单元需要再通过 tf.nn.dynamic_rnn 函数来封装成 LSTM。tf.nn.rnn_cell.LSTMCell 函数的参数及返回值如表 3.3 所示。

表 3.3 tf.nn.rnn_cell.LSTMCell 参数及返回值

变量标识	变量作用	变量说明
num_units	函数参数	隐藏层节点数
cell	函数返回值	返回一个神经网络单元

使用 tf.nn.rnn_cell.GRUCell 函数来构建一个 GRU 的神经网络单元，得到的神经网络单元需要再通过 tf.nn.dynamic_rnn 函数来封装成 GRU。tf.nn.rnn_cell.GRUCell 函数的参数及返回值如表 3.4 所示。

表 3.4 tf.nn.rnn_cell.GRUCell 参数及返回值

变量标识	变量作用	变量说明
num_units	函数参数	隐藏层节点数
cell	函数返回值	返回一个神经网络单元

使用 tf.nn.rnn_cell.MultiRNNCell 函数来构建一个 DRNN 的聚合神经网络单元，其中 cells 中的基础神经网络单元可以是标准 RNN、LSTM、GRU 等，得到的神经网络单元需要再通过 tf.nn.dynamic_rnn 函数来封装成 DRNN。tf.nn.rnn_cell.MultiRNNCell 函数的参数及返回值如表 3.5 所示。

表 3.5 tf.nn.rnn_cell.MultiRNNCell 参数及返回值

变量标识	变量作用	变量说明
cells	函数参数	按顺序组成的 RNNCells 的列表
state_is_tuple	函数参数	若为 True，接受和返回的状态是 n 个元组，n=len(cells)，若为 False，状态沿列轴串联
mul_cells	函数返回值	返回一个聚合神经网络单元

BRNN 由两个 RNN 单元连接而成，一个隐藏层向前推算，另一个隐藏层向后推算，其中 RNN 单元可以是标准 RNN、LSTM、GRU 等。使用 tf.nn.bidirectional_dynamic_rnn 函数来构建一个 BRNN，函数返回值 outputs 和 states 均是长度为2的元组，需要将它们合并成一个整体才方便进行模型训练。利用 tf.concat 函数可以将 outputs 合并成 output_rnn，利用 tf.concat 函数和 tf.nn.rnn_cell.LSTMStateTuple 函数可以将 states 合并成 outputs_final_state。tf.nn.bidirectional_dynamic_rnn 函数的参数及返回值如表 3.6 所示。

表 3.6 tf.nn.bidirectional_dynamic_rnn 参数及返回值

变量标识	变量作用	变量说明
fw_cell	函数参数	向前推算的神经网络单元
bw_cell	函数参数	向后推算的神经网络单元
inputs	函数参数	输入的训练数据
initial_state_fw	函数参数	向前推算的神经网络单元的初始状态
initial_state_bw	函数参数	向后推算的神经网络单元的初始状态
dtype	函数参数	数据类型
outputs	函数返回值	最后的输出结果，类型为元组，包含向前和向后
states	函数返回值	最后一个 cell 的状态，类型为元组，包含向前和向后

（3）预测模型训练

预测模型训练的具体训练过程如图 3.3 所示，预测模型训练时的一些重要参数及含义如表 3.7 所示。在本文中，所有的神经网络模型除网络结构不同外，模型训练过程大体上是一致的，即先获取训练所用的数据集，然后调用构造好的神经网络结构，最后进行循环训练，满足一定要求时退出循环，此次训练完毕。

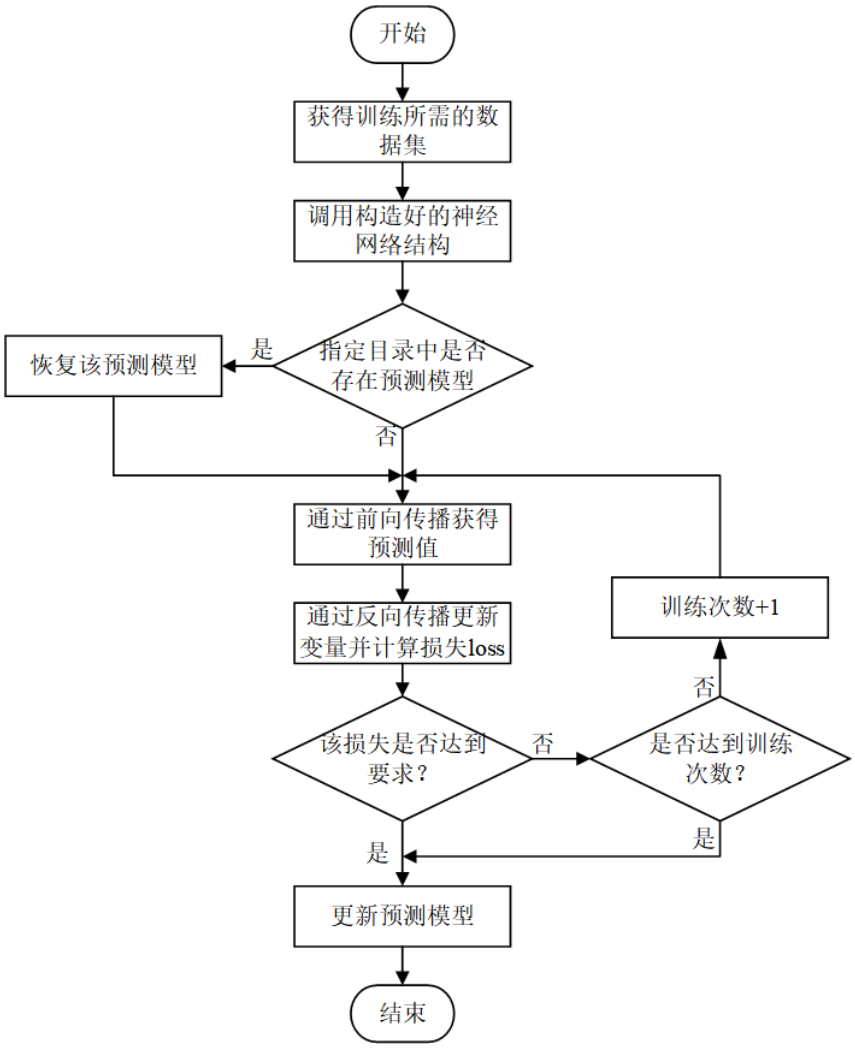

图 3.3　预测模型训练流程图

表 3.7　预测模型训练时的参数

参数名称	参数含义	设置值
input_size	输入维度	1
output_size	输出维度	1
rnn_unit	隐藏层节点	20
lr	初始学习率	0.001
batch_size	每次训练的一个批次的大小	25
time_step	时间步，用前 time_step 步来预测下一步	20
predict_step	预测步，预测 predict_step 步后的值	20
train_time	所有数据的训练轮次	10000

3.2.2 负载预测子模块实现

负载预测的具体过程如图 3.4 所示。在线上环境中，首先从输入队列中取一个时间步长的负载数据，然后读取训练好的预测模型，最后将数据输入到预测模型中获得负载预测值。

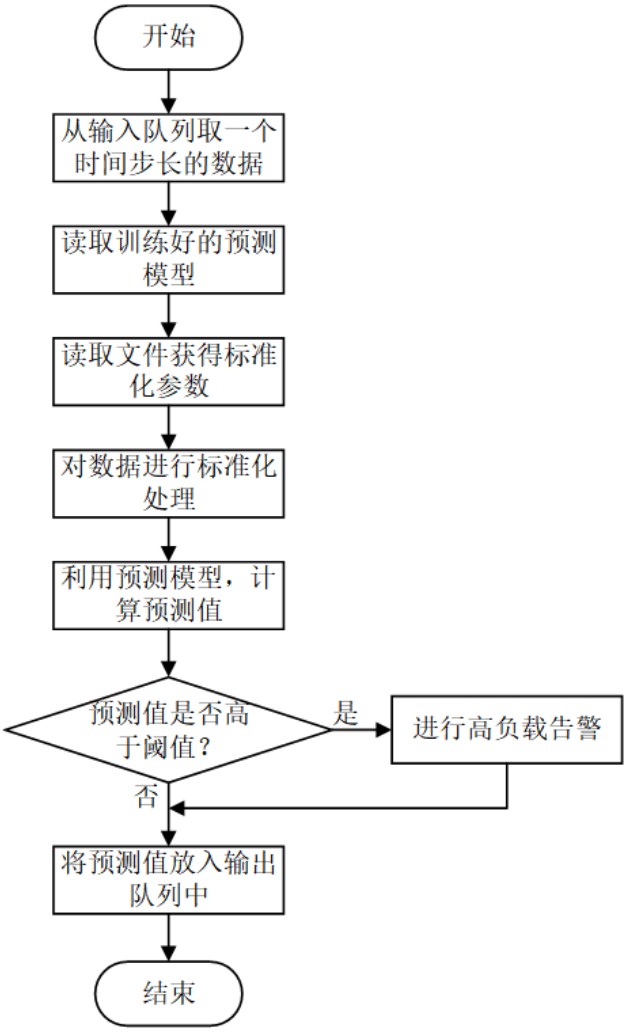

图 3.4 负载预测流程图

使用阈值来判断预测值是否为高负载，阈值的计算方法如下：使用一个变量来统计负载请求的平均值，该变量会在负载数据进入到输入队列前进行统计，然后设定高于负载请求平均值的 1.5 倍或者高于 10000 即为高负载，会进行高负载告警及生成调度指令。输入队列中的负载数据的数据结构如表 3.8 所示，输出队列中的负载数据的数据结构如表 3.9 所示。

表 3.8　输入队列中的数据结构

字段	类型	说明
serverIP	string	所属的服务器 IP 地址
serverName	string	所属的服务器名称
requestLoad	int	服务器中键值存储系统每分钟接收到的负载请求量
timeStamp	float	该负载数据进入队列的时间戳

表 3.9　输出队列中的数据结构

字段	类型	说明
serverIP	string	所属的服务器 IP 地址
serverName	string	所属的服务器名称
requestLoad	int	服务器中键值存储系统每分钟接收到的负载请求量
requestLoadPredict	int	服务器中键值存储系统负载请求的预测值
timestamp	float	该负载数据进入队列的时间戳

3.2.3　动态训练子模块实现

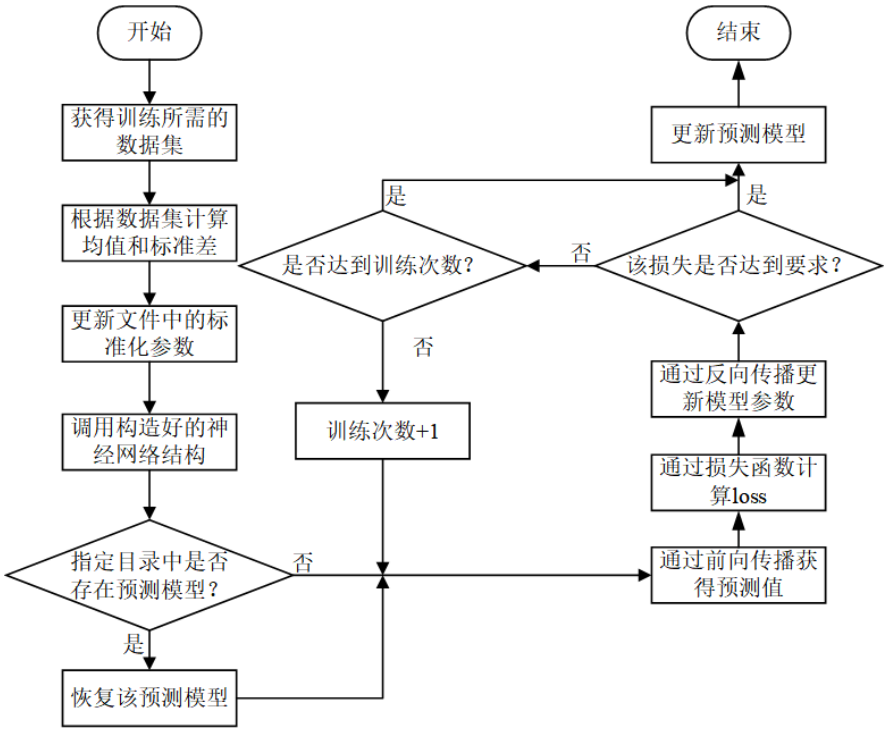

图 3.5　预测模型动态训练流程图

预测模型动态训练的具体过程如图 3.5 所示。线下模型训练的过程与线上动态训练的过程大致一致，前者数据来源是典型的互联网应用数据集，后者数据来源是智能调度模块所收集的键值存储系统的真实负载请求。在动态训练时需要更新标准化参数，以提高负载预测的准确度，数据集的均值和标准差使用 numpy.mean 函数和 numpy.std 函数计算得到。

3.3 基础功能模块实现

基础功能模块能够为系统的正常运行提供基础功能支持，该模块主要由三个子模块组成：系统配置子模块、图像显示子模块和服务器通信子模块，下面将详细介绍这些子模块是如何实现的。

3.3.1 系统配置子模块实现

服务器自动配置的具体过程如图 3.6 所示。使用 UDP 协议中的广播功能来实现服务器的自动配置。系统会在后台线程监听特定端口，当有报文送达时，对该报文进行解析，若该报文格式符合要求，则将该报文中的附带的 IP 地址添加到系统中。其中，socket 函数中的 type 参数需设置为 socket.SOCK_DGRAM 来返回 UDP 类型的套接字；setsockopt 函数中的 optname 参数需设置为 socket.SO_BROADCAST 来允许发送广播。

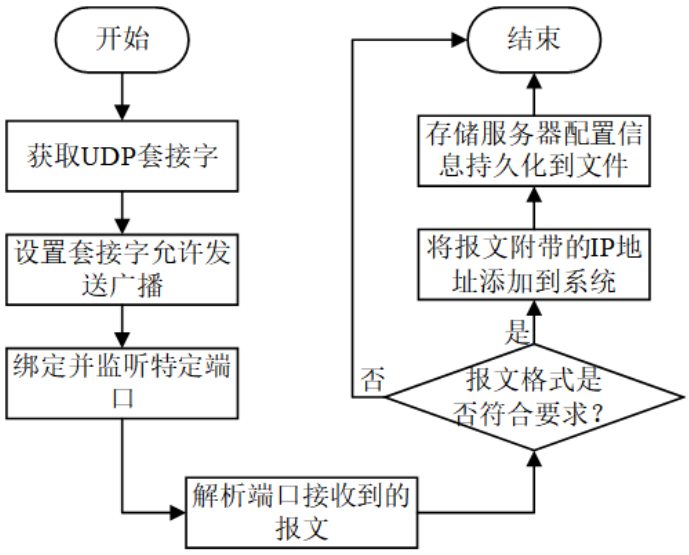

图 3.6 服务器自动配置流程图

服务器手动配置的具体过程如图 3.7 所示。相比于自动配置，服务器的手动配置的实现就比较简单。手动输入服务器 IP 地址、名称等信息，在系统验证通过后则将 IP 地址等信息添加到系统中。其中，使用正则表达式来判断输入信息是否符合格式要求；socket 函数中的 type 参数需设置为 socket.SOCK_STREAM 来返回 TCP 类型的套接字；settimeout 函数中的 value 参数设置为 2，表示套接字的超时阈值为 2 秒，超出这个时间则会抛出超时异常。

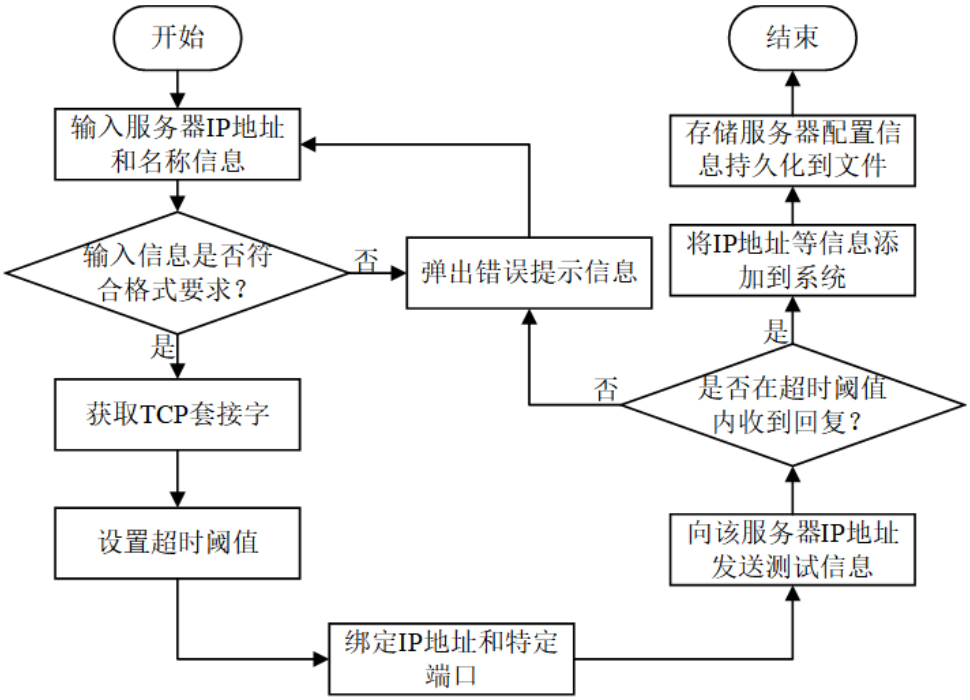

图 3.7　服务器手动配置流程图

内存中所保存的服务器信息始终是最新的，因此对服务器信息的查询、修改和删除操作可以转化为对内存中保存服务器信息的数据结构进行查询、修改和删除操作，然后再将该信息持久化到配置文件中即可。在本文中，使用一个二维列表来保存服务器信息，列表中的每个元素是一个长度为 2 的列表，存放着每个服务器的 IP 地址和名称信息，使用 json.dump 函数可以将内存中的数据结构持久化到文件中，使用 json.load 函数可以将文件中的信息恢复成内存中的数据结构。

3.3.2　图像显示子模块实现

该子模块负责整个系统的界面显示，包括服务器配置界面、负载显示界面等。在

本文中，使用 PyQt5 这个第三方库来实现系统的布局管理、对话框以及各种功能控件等，由于界面的具体实现比较繁琐，这里不再赘述，所使用 PyQt5 中的一些控件如表 3.10 所示：

表 3.10　所使用 PyQt5 中的一些控件

控件名称	控件说明
QPushButton	常见按钮，可设置图标，可控制按钮可用与不可用
QComboBox	下拉列表框，可以获得当前所选项和索引，可用于限制输入
QLineEdit	单行文本控件，用于文本输入
QMessageBox	用于提示信息, 如警告、询问和严重出错等
QTabWidget	一个选项卡对应一个页面区域，单击选项卡查看各个页面
QTableWidget	表格控件，用于绘制表格

负载显示界面需显示云服务器中键值存储系统的实时负载请求变化情况和预测负载变化情况，以分钟作为基本单位。在本文中，使用 pyecharts 这个第三方库来绘制负载请求曲线图，pyecharts 是在 Echarts 库的基础上使用 Python 语言进行了封装处理，主要用于数据可视化，内置众多图表绘制的接口。pyecharts 中包含的常见图表如表 3.11 所示：

表 3.11　pyecharts 中包含的常见图表

对象名称	对应图表
Bar	柱状图
Scatter	散点图
Line	折线图
Pie	饼图
Rader	雷达图
Boxplot	箱形图

3.3.3　服务器通信子模块实现

服务器通信的具体过程如图 3.8 所示。该子模块能够向部署在云服务器键值存储系统中的智能调度模块请求负载数据和发送调度指令。负载数据是每分钟向服务器

请求一次，而调度指令则是预测到即将出现高负载时才会发送。其中数据请求类型有两种，类型 1 中仅包含实时负载请求数据，类型 2 中包含实时负载请求数据和动态训练负载数据。动态训练不需要每时每刻都进行，因此本文设定是每两小时请求一次动态训练负载数据，数据请求类型由该子模块中的计时器来判断。其中，socket 函数中的 type 参数需设置为 socket.SOCK_STREAM 来返回 TCP 类型的套接字；index 即为该子模块的计时器，index 的数据结构类型为字典，字典中的键为服务器的 IP 地址，字典中的值初始值为 0，每请求一次数据该值就会加 1，循环周期为 2 小时，即 120 分钟。

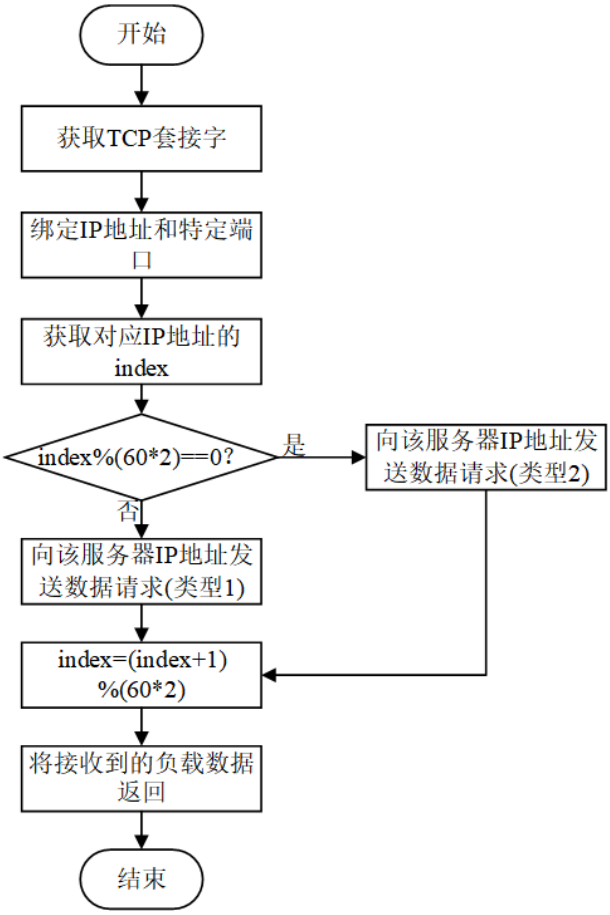

图 3.8　服务器通信流程图

负载数据在按照 TCP 协议进行传输前，需使用 json.dumps 函数将其转化为字符串以方便传输，在接收到负载数据后，需使用 json.loads 函数将其还原为原来的数据结构类型。接收的负载数据的数据结构类型为字典，当数据请求为类型 1 时，字典的

键仅为“realtime_request_load_data”，当数据请求为类型 2 时，字典的键为“realtime_request_load_data”和“dynamic_training_load_data”，字典的值即是对应服务器中键值存储系统的负载请求数据。

使用字典来存放实时负载请求数据和动态训练负载数据，字典的键为服务器的 IP 地址，字典的值为对应的负载数据的列表，存放动态训练负载数据的列表长度比存放实时负载请求数据的列表长度更长。

3.4 智能调度模块实现

智能调度模块能够为系统提供实时的负载数据，该模块主要由两个子模块组成：数据收集子模块和弹性调度子模块，下面将详细介绍这些子模块是如何实现的。

3.4.1 数据收集子模块实现

数据收集的具体过程如图 3.9 所示。该子模块所收集的实时负载请求数据和动态训练负载数据都以分钟作为基本单位。在 RocksDB 对 SSTable 数据文件进行读写操作前，编写相关统计函数即可收集 RocksDB 的实时负载请求数据。

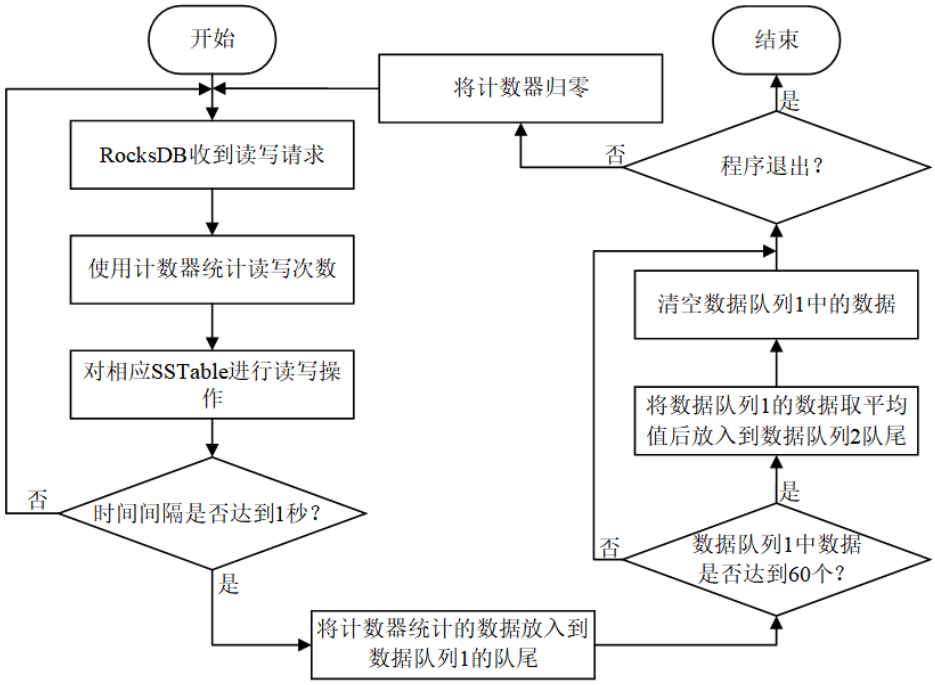

图 3.9 数据收集流程图

后台线程每秒钟收集一次键值存储系统的负载请求数据并存放与数据队列 1 中，

当数据队列 1 中的数据满 60 个后，将其转化为每分钟的负载请求数据并清空数据队列 1，然后将每分钟的负载请求数据存放在数据队列 2 中，数据队列 1 和数据队列 2 的数据结构类型均为列表。系统所需的实时负载请求数据即为数据队列 2 的队尾数据，动态训练负载数据即为数据队列 2 的所有数据。

数据发送的具体过程如图 3.10 所示。该子模块根据数据请求的类型发送相应类型的负载请求数据，当接收到实时负载数据的请求时，发送实时负载请求数据，当接收到动态训练负载数据的请求时，发送实时负载请求数据和动态训练负载数据。其中，socket 函数中的 type 参数需设置为 SOCK_STREAM 来返回 TCP 类型的套接字；发送动态训练负载数据的条件为数据队列 2 的长度大于等于 300，即 5 个小时的负载请求数据，而数据队列 2 的长度上限设置为 1440，即最多存储 24 个小时的负载请求数据；使用 json.dumps 函数将数据结构转化为字符串以方便传输。

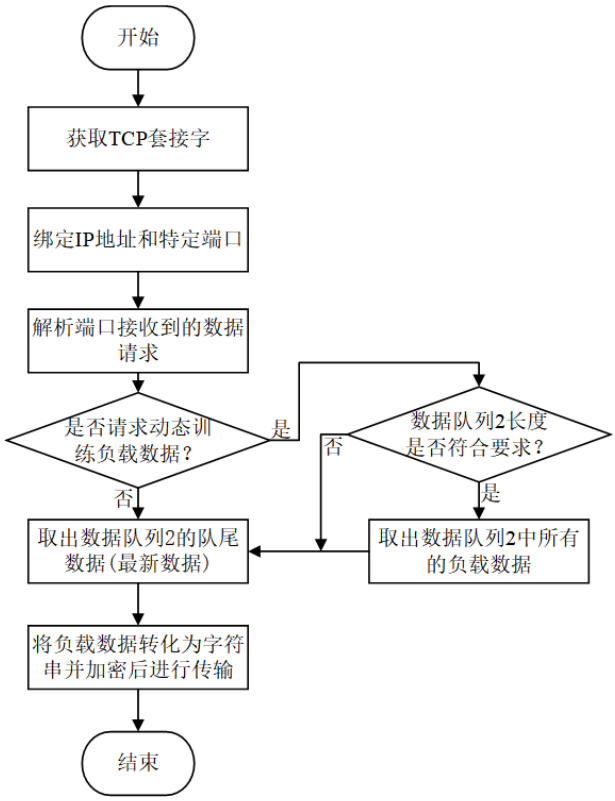

图 3.10　数据发送流程图

另外，该子模块需在后台线程广播所在服务器的 IP 地址，来实现基础功能模块的服务器自动配置功能。socket 函数中的 type 参数需设置为 SOCK_DGRAM 来返回

UDP 类型的套接字，setsockopt 函数中的 optname 参数需设置为 SO_BROADCAST 来允许发送广播，广播的具体内容为系统的英文缩写“IESS”加上 IP 地址的字符串，中间以下划线作为分隔符，便于基础功能模块判断该报文是否属于 IESS，以便进行后续操作。

3.4.2 弹性调度子模块实现

弹性调度的具体过程如图 3.11 所示。该子模块在接收到 IESS 发送的调度指令后，首先计算云服务器键值存储系统中每个 SSTable 数据文件的热度值，然后确定每个磁盘中需要进行数据迁移的 SSTable 数据文件，再申请扩充新的磁盘，最后将需要进行迁移的 SSTable 数据文件迁移至新扩充的磁盘中。以上操作均在前台负载高峰来临前完成，可以有效避免键值存储系统因数据迁移而导致的性能下降的问题，实现键值存储系统对存储资源的弹性调度。

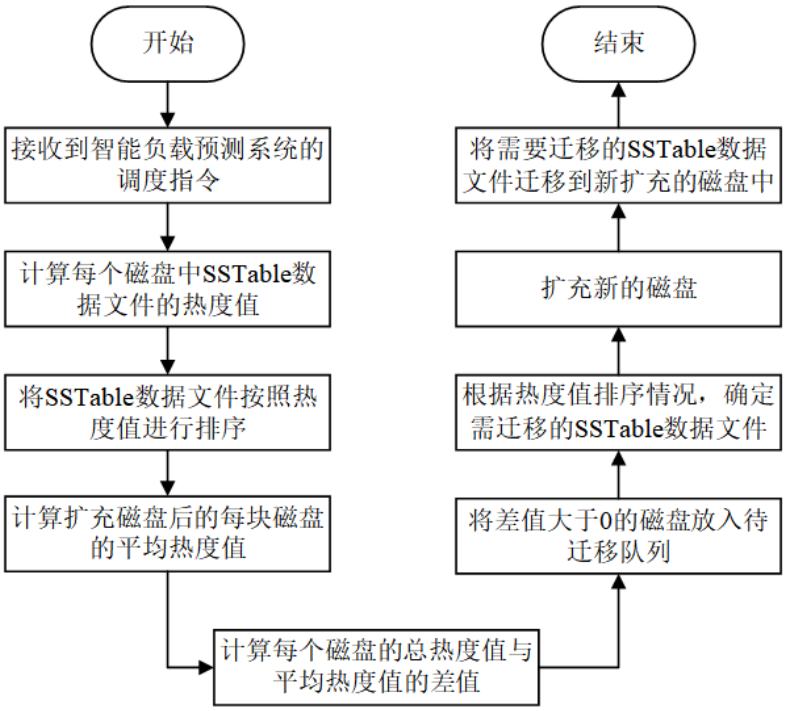

图 3.11 弹性调度流程图

其中，热度统计函数在 RocksDB 对 SSTable 数据文件进行访问前调用，所有 SSTable 数据文件的访问记录都被保存在一个 map 中，map 的键为 SSTable 数据文件的 id，map 的值是一个列表，存放每次访问的时间戳。列表的长度即为该 SSTable 数

据文件被访问的总次数 n，根据每次访问的时间戳来计算出最近一段时间Δt内被访问的次数 m，根据热度值计算公式（2.6）计算出该 SSTable 数据文件在当前时刻的热度值大小。本文在数据迁移上进行了简单处理，在数据迁移时，只需将待迁移的 SSTable 数据文件复制到新的存储资源中，而不需删除原来的 SSTable 数据文件。当键值存储系统通过阈值判断处于低负载时刻时，会直接释放掉前面扩充的存储资源，而不需要再进行数据迁移操作。

3.5 本章小结

本章主要介绍 IESS 是如何实现的。首先给出了 IESS 的整体架构图，然后分别介绍组成系统的智能预测模块、基础功能模块和智能调度模块以及它们的子模块是如何实现的，详细地介绍其中涉及的数据结构、函数参数及返回值、功能算法实现以及主要流程实现。

4 测试与结果分析

经过第二章的设计与第三章的实现，本文逐步实现了 IESS，该系统可以基于负载预测来实现键值存储系统对存储资源的弹性调度。本章内容主要是搭建一个测试环境，分别对 IESS 的负载预测与弹性调度这两项功能进行测试，并对测试结果进行详细分析，以验证测试结果是否符合设计预期。

4.1 测试方案

整个测试环节分为两个部分：

（1）负载预测模型测试

该测试在线下环境中进行，训练数据采用的是典型的互联网应用数据集，包括出行服务和点餐服务这两种典型互联网应用的负载数据。原始数据集无法直接使用，需要进行数据处理后得到以分钟为单位的负载数据才可以用于预测模型的训练。

本文使用出行服务和点餐服务这两种典型互联网应用的数据集，分别对标准 RNN 模型、LSTM 模型、GRU 模型、DRNN 模型和 BRNN 模型进行重复迭代训练，以提高负载预测的准确度。本文采用 MAE、RMSE 和 R2_Score 等多种评估方式对每个 RNN 模型的预测准确度进行评估。

（2）键值存储系统智能弹性调度测试

该测试在线上环境中进行，IESS 会根据云服务器中键值存储系统的实时负载请求数据来动态训练预测模型，以提高负载预测的准确度。云服务器中的键值存储系统会根据 IESS 发送的调度指令，在负载请求高峰期来临前进行调度操作，以避免因数据迁移而产生的负载对键值存储系统性能产生影响。

在出行服务和点餐服务这两种典型互联网应用负载的场景下，分别采取无调度、被动调度和弹性调度这三种调度策略来对云服务器的存储资源进行调度。在相同负载变化场景中，测试在三种调度策略下键值存储系统的每秒查询率（Queries Per Second，QPS）和成本开销的差异，并根据测试结果得出相应的结论。

4.2 负载预测模型测试

该测试采用 MAE、RMSE 和 R2_Score 等多种评估方式在相同应用负载环境下对每个 RNN 模型的预测准确度进行评估，根据测试结果，确定预测效果最好的 RNN 模型。

4.2.1 测试环境

该测试在线下环境中进行，测试所用计算机的相关配置如表 4.1 所示：

表 4.1 测试所用计算机相关配置

操作系统	Windows 10
处理器	AMD Ryzen 7 4700U 2.00 GHz * 8 Core
主存	RAM 16GB
SSD	WDC PC SN730 SDBPNTY-512G-1027 512GB

使用出行服务和点餐服务这两种典型互联网应用的负载数据作为数据集，从数据集中截取三天的负载数据用来训练模型，将前两天的负载数据作为训练集，第三天的数据作为测试集，如图 4.1 所示。训练集用于重复训练预测模型，测试集用于评估预测模型进行负载预测的准确度。

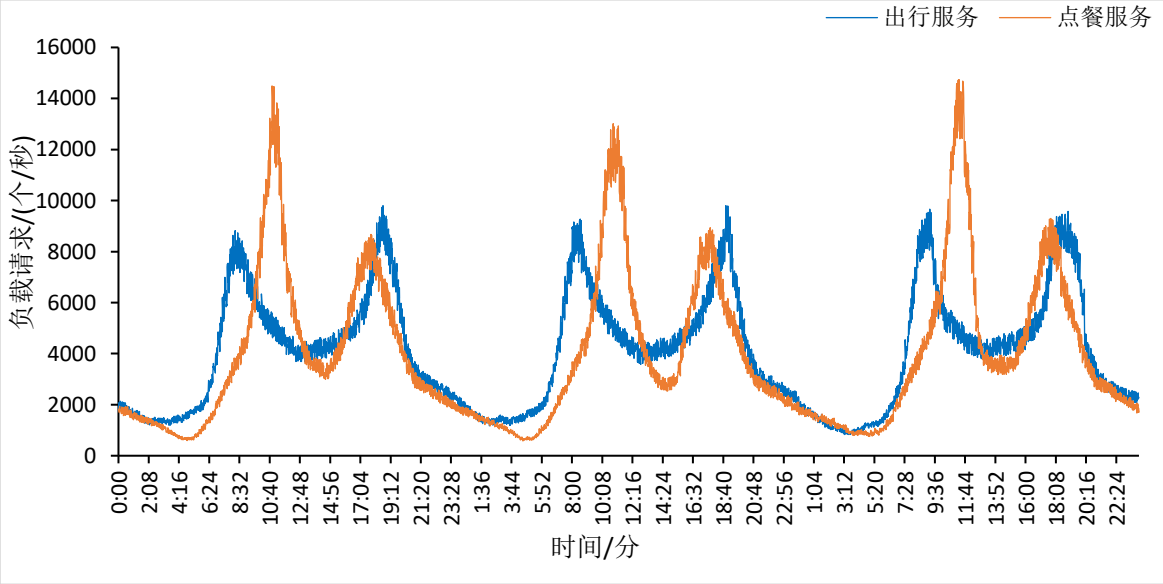

图 4.1 典型互联网应用的负载数据示意图

4.2.2 测试结果及分析

（1）使用出行服务应用数据集的测试

在使用训练集进行模型训练时，若满足以下任意条件：（1）损失函数的值小于 1e-5；（2）所有数据的训练轮次达到 2000 次，则表明预测模型具有较高的预测精度，此次模型训练结束。所有 RNN 模型在训练过程中的表现以及在测试集上的表现统计如表 4.2 所示。

表 4.2　所有预测模型的训练及预测表现

模型名称	训练时间	训练轮次	损失 loss	MAE	RMSE	误差百分比	R2_Score
标准 RNN	707.1s	2000	5.46e-4	479855	21186	11.69%	92.23%
LSTM	556.7s	723	7.21e-6	358205	14650	8.73%	96.28%
GRU	431.4s	613	8.47e-6	382023	16733	9.31%	95.15%
DRNN_RNN	961.4s	1931	9.53e-6	434115	18027	10.58%	94.37%
DRNN_LSTM	747.6s	543	6.14e-6	208984	9932	**5.09%**	**98.29%**
DRNN_GRU	503.1s	447	7.88e-6	345105	14216	8.41%	96.41%
BRNN_RNN	695.4s	2000	4.78e-4	460913	20337	11.23%	92.84%
BRNN_LSTM	549.7s	667	6.98e-6	337946	14223	8.24%	96.49%
BRNN_GRU	398.3s	514	7.93e-6	370957	16213	9.04%	95.36%

从表 4.2 可以看出，在单层 RNN 模型中，预测效果最好的是 LSTM 模型，误差百分比和 R2_Score 分别为 8.72%和 96.28%。而在 DRNN 模型中，预测效果最好的是 DRNN_LSTM 模型，误差百分比和 R2_Score 分别为 5.09%和 98.29%。在 BRNN 模型中，其预测效果与对应的单层网络的预测模型差异不大，预测效果最好的是 BRNN_LSTM 模型，误差百分比和 R2_Score 分别为 8.24%和 96.49%。

综合来看，预测效果最好的是 DRNN_LSTM 模型，它在测试集上的表现如图 4.2 所示，其中横纵坐标分别表示时间和负载请求大小，红色曲线表示负载真实值，蓝色虚线表示预测模型输出的负载预测值。

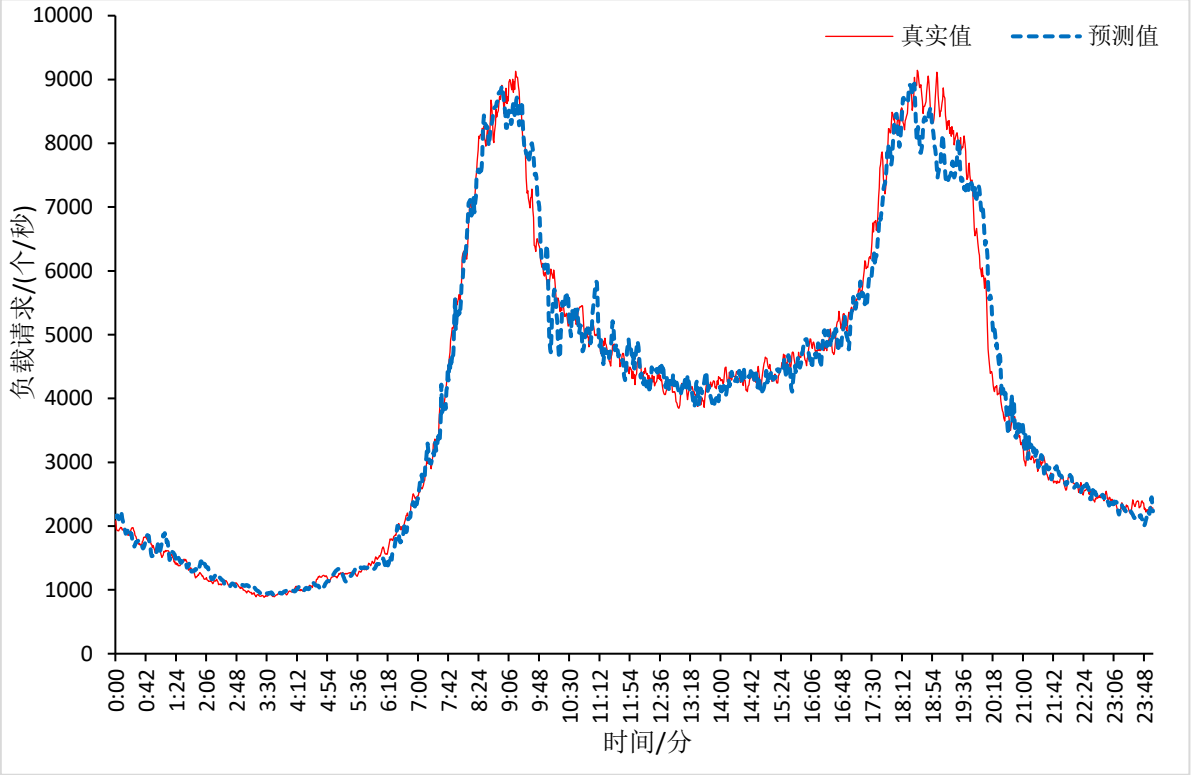

图 4.2　DRNN_LSTM 模型预测效果图

从图 4.2 可以看出，在出行应用的负载数据集中，负载高峰期主要集中在早高峰时间和晚高峰时间，两个高峰期的峰值区别不大，但是晚高峰持续时间更长，这是因为某些公司可能存在加班现象。代表负载预测值的蓝色虚线虽然有些许波动，但基本上与代表负载真实值的红色曲线重合在一起，二者的总体变化趋势是一致的。DRNN_LSTM 模型在测试集上的误差百分比和 R2_Score 分别为 5.09%、98.29%，这也说明该预测模型对于出行服务这种应用场景的负载变化具有较好的预测效果。

（2）使用点餐服务应用数据集的测试

在使用训练集进行模型训练时，若满足以下任意条件：（1）损失函数的值小于 1e-5；（2）所有数据的训练轮次达到 2000 次，则表明预测模型具有较高的预测精度，此次模型训练结束。所有 RNN 模型在训练过程中的表现以及在测试集上的表现统计如表 4.3 所示。

表 4.3　所有预测模型的训练及预测表现

模型名称	训练时间	训练轮次	损失 loss	MAE	RMSE	误差百分比	R2_Score
标准 RNN	721.2s	2000	6.70e-4	589127	24326	14.13%	93.14%
LSTM	563.1s	668	7.79e-6	368618	20425	8.84%	95.54%
GRU	335.7s	561	8.31e-6	560519	22301	13.44%	94.69%
DRNN_RNN	930.7s	1871	8.57e-6	568697	22649	13.64%	94.02%
DRNN_LSTM	703.4s	587	6.23e-6	300200	14546	**7.20%**	**97.74%**
DRNN_GRU	448.3s	458	8.21e-6	431109	20884	10.34%	95.67%
BRNN_RNN	701.9s	2000	5.64e-4	564528	22887	13.87%	93.58%
BRNN_LSTM	503.2s	632	7.59e-6	419454	20008	10.06%	95.72%
BRNN_GRU	320.4s	523	8.03e-6	524919	21884	12.59%	94.81%

从表 4.3 可以看出，在单层 RNN 模型中，预测效果最好的是 LSTM 模型，误差百分比和 R2_Score 分别为 8.84%和 95.54%。而在 DRNN 模型中，预测效果最好的是 DRNN_LSTM 模型，误差百分比和 R2_Score 分别为 7.20%和 97.74%。在 BRNN 模型中，其预测效果与对应的单层网络的预测模型差异不大，预测效果最好的是 BRNN_LSTM 模型，误差百分比和 R2_Score 分别为 10.06%和 95.72%。

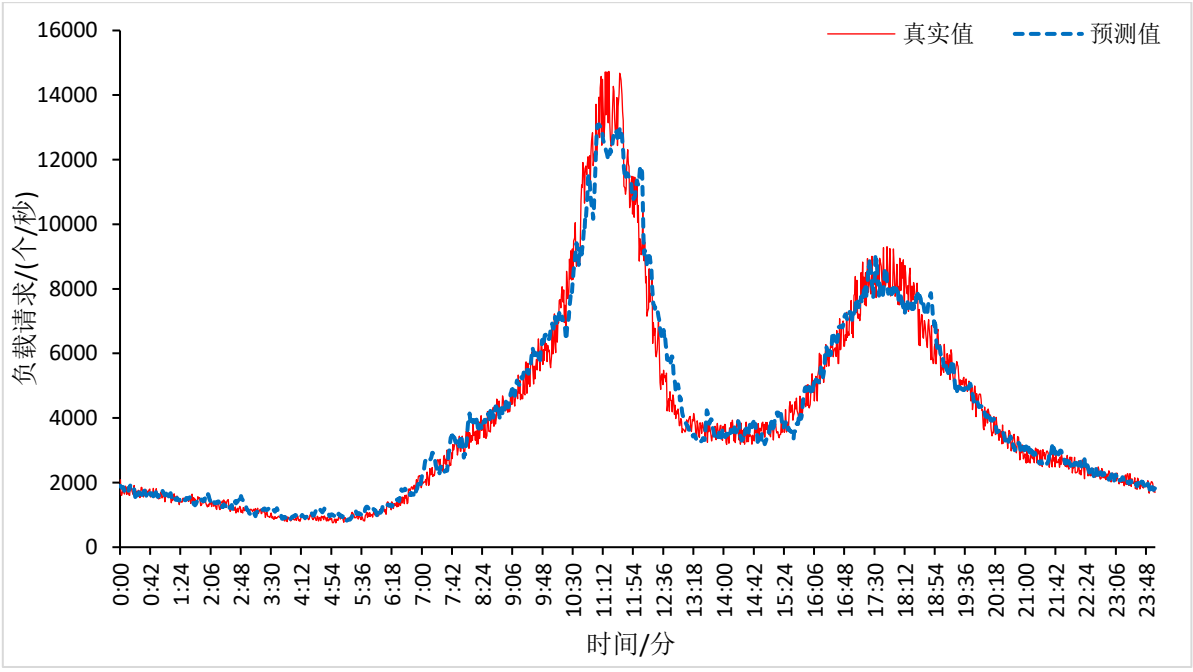

图 4.3　DRNN_LSTM 模型预测效果图

综合来看，预测效果最好的是 DRNN_LSTM 模型，它在测试集上的表现如图 4.3 所示，其中横纵坐标分别表示时间和负载请求大小，红色曲线表示负载真实值，蓝色虚线表示预测模型输出的负载预测值。

从图 4.3 可以看出，在点餐服务应用的负载数据集中，负载高峰期主要集中在中午和晚上，且中午的高峰期明显高于晚上的高峰期，这是因为人们更倾向于回家吃晚餐。代表负载预测值的蓝色虚线虽然有些许波动，但基本上与代表负载真实值的红色曲线重合在一起，二者的总体变化趋势是一致的。DRNN_LSTM 模型在测试集上的误差百分比和 R2_Score 分别为 7.20%、97.74%，这也说明该预测模型对于点餐服务这种应用场景的负载变化具有较好的预测效果。

4.2.3 测试小结

使用出行服务应用和点餐服务应用的数据集对所有预测模型进行训练和测试，从表 4.2 和表 4.3 中可以发现，在单层 RNN 模型中，标准 RNN 模型简单的网络结构决定了它拥有非常快的训练速度，但预测效果往往不尽人意，训练时会因损失 loss 较大而不能提前结束训练；相比于 LSTM 模型，GRU 模型的网络结构更加精简，因此训练速度更快，但在预测效果上稍稍逊色于 LSTM 模型。在 DRNN 模型中，将普通的单层网络的进行叠加来构建更加复杂的网络结构，因而其预测效果更优于单层网络结构的模型，与单层网络中的规律一致，预测效果最好的是 DRNN_LSTM 模型。在 BRNN 模型中，其预测效果与对应的单层网络的预测模型差异不大，这可能是因为 BRNN 模型并不适用于负载预测这种场景（BRNN 模型更适用于语音识别）。

测试结果表明，DRNN_LSTM 模型的预测效果最好，对于出行服务和点餐服务这两种应用场景的负载变化都具有较好的预测效果。

4.3 键值存储系统智能弹性调度测试

由 4.2 小节的测试结果可知，DRNN_LSTM 模型的预测效果最好，因此采用 DRNN_LSTM 模型作为预测模型，测试键值存储系统智能弹性调度的效果，并在相同应用负载场景下，与无调度与被动调度作对比，得出相应结论。

4.3.1 测试环境

该测试在线上环境中进行，示意图如图 4.4 所示。IESS 部署在主机与云服务器上，基于负载预测来生成调度指令，并将调度指令发送给云服务器中的键值存储系统，键值存储系统根据调度指令来对存储资源进行弹性调度操作。

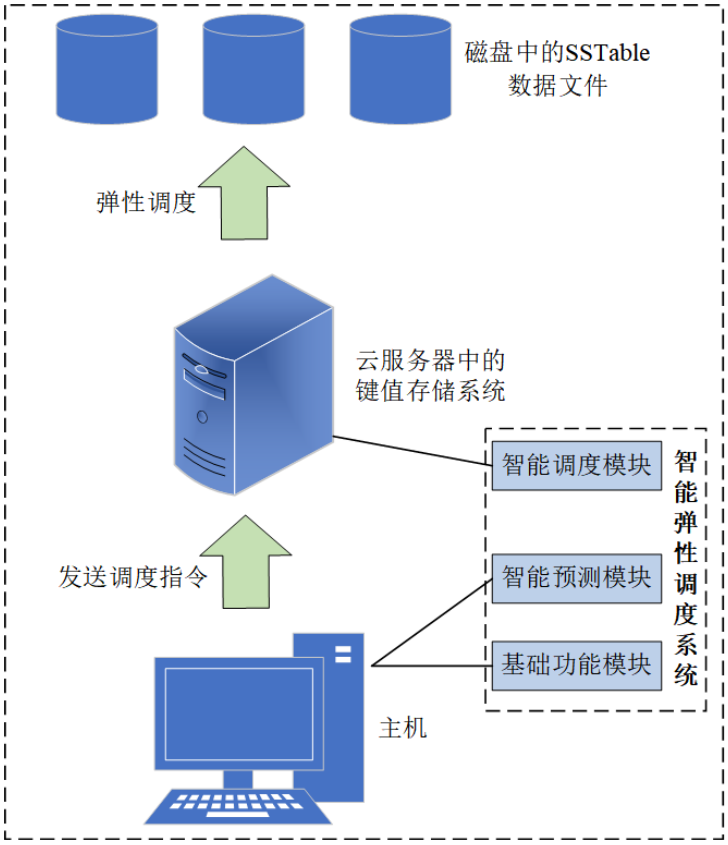

图 4.4 测试环境示意图

测试所用云服务器的相关配置如表 4.4 所示。此外，还需要在云服务器上安装 db_bench 测试工具，该工具可以自定义各种负载测试与压力测试来模拟服务器真实负载情况。

表 4.4 测试所用服务器相关配置

操作系统	Ubuntu
内核版本	Linux 5.13.0
处理器	Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz *8 Core
主存	DDR3REG 32GB
SSD	Intel SSDSC2BB800G7 800GB

4.3.2 测试结果及分析

（1）出行服务应用负载场景下的测试

使用 db_bench 测试工具来模拟出行服务应用的负载场景，对云服务器中的 RocksDB 进行压力测试，当 RocksDB 的负载请求变化处于一个稳定的范围后，开启 IESS，其中智能预测模块需在线上进行动态训练，以确保负载预测的准确度。

在相同负载场景下，采取无调度、被动调度和弹性调度这三种调度策略来对云服务器的存储资源进行调度，RocksDB 在这三种调度策略下 QPS 的变化如图 4.5 所示：

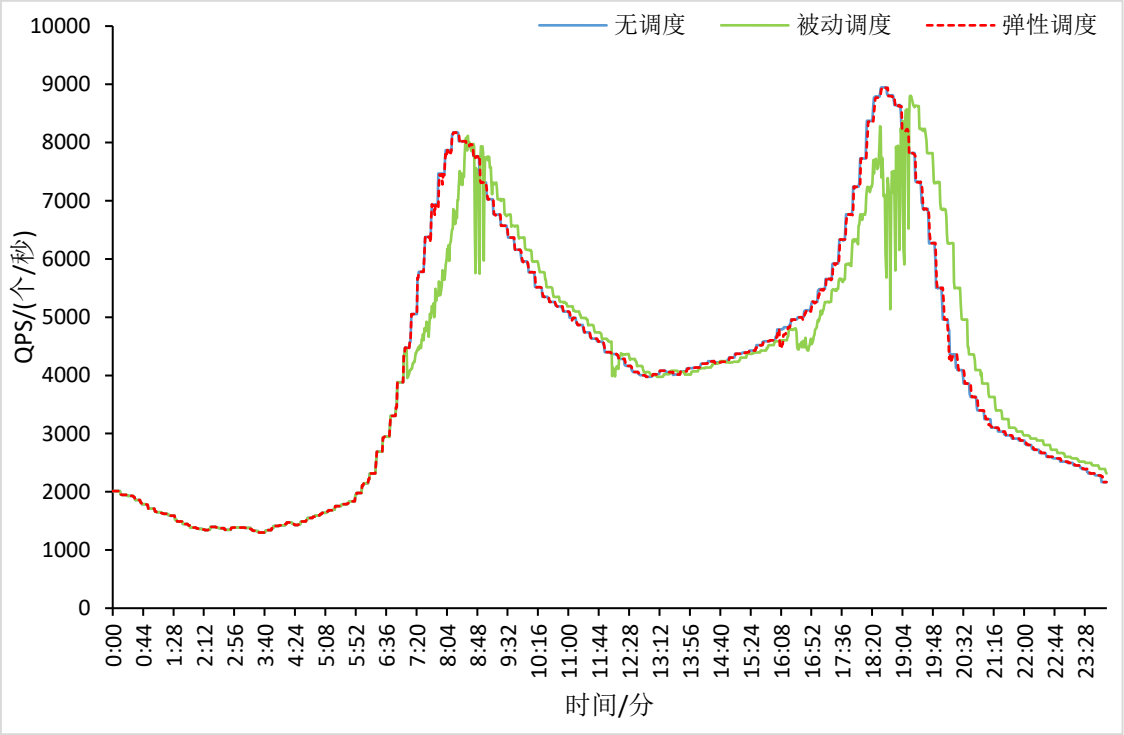

图 4.5　三种不同调度策略下 QPS 变化图

从图 4.5 中可以看到，表示无调度 QPS 的蓝色曲线与表示弹性调度 QPS 的红色虚线基本上重合在一起，而表示被动调度 QPS 的绿色曲线则有一定的滞后性。

QPS 越高表示性能越好，三种调度策略下 RocksDB 的平均 QPS 如图 4.6 所示，可以看到，弹性调度的平均 QPS 略低于无调度的平均 QPS，说明弹性调度的性能与无调度的性能很接近，而被动调度的平均 QPS 最低，说明被动调度的性能较差。

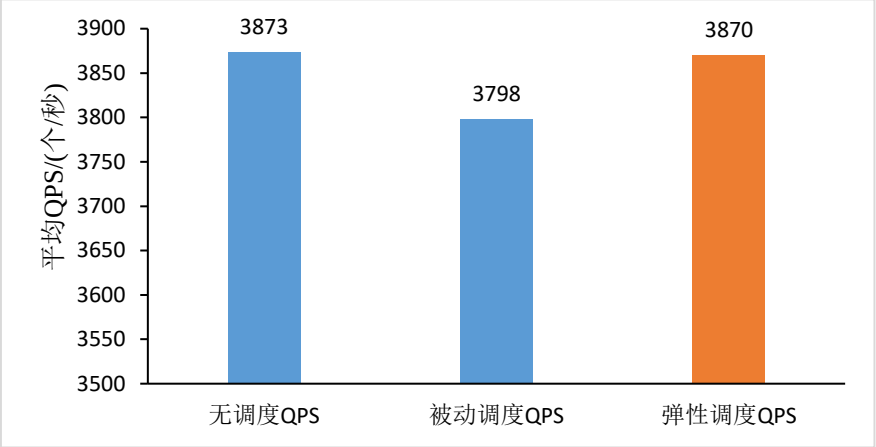

图 4.6　三种不同调度策略下平均 QPS

RocksDB 会根据 IESS 生成的调度指令对存储资源进行弹性调度，磁盘 IOPS（Input/Output Per Second）变化如图 4.7 所示：

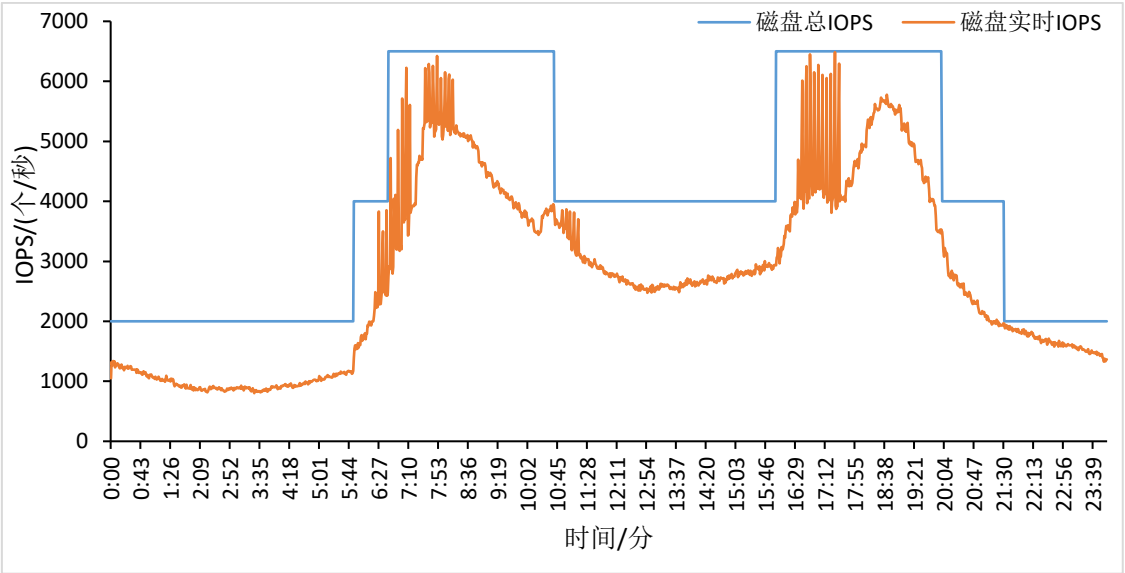

图 4.7　磁盘 IOPS 变化图

在图 4.7 中，蓝色曲线表示磁盘总 IOPS，橙色曲线表示磁盘实时 IOPS，可以看到，在前台负载增加时，RocksDB 会提前扩充存储资源，从而增加磁盘总 IOPS，而数据迁移的负载会使磁盘实时 IOPS 产生不规则波动，在前台负载下降时，RocksDB 会缩减存储资源，从而减少磁盘总 IOPS，由于不需要再进行数据迁移，因而磁盘实时 IOPS 不会产生不规则波动。

三种调度策略下，一天所需的存储资源如表 4.5 所示：

表 4.5　三种调度策略下一天所需的存储资源

调度策略	一天所需的存储资源
无调度	100GB 6500IOPS io2 24h
被动调度	100GB 2000IOPS io2 24h 50GB 2000IOPS io2 15h 50GB 2500IOPS io2 8h
弹性调度	100GB 2000IOPS io2 24h 50GB 2000IOPS io2 15h30min 50GB 2500IOPS io2 8h30min

云服务器中的存储资源价格为每 24 小时每 100GB 的费用为 0.4167 美元，每 24 小时每 1000IOPS 的费用为 2.1667 美元，由此可以计算表 4.5 中所需的存储资源的成本开销，如图 4.8 所示：

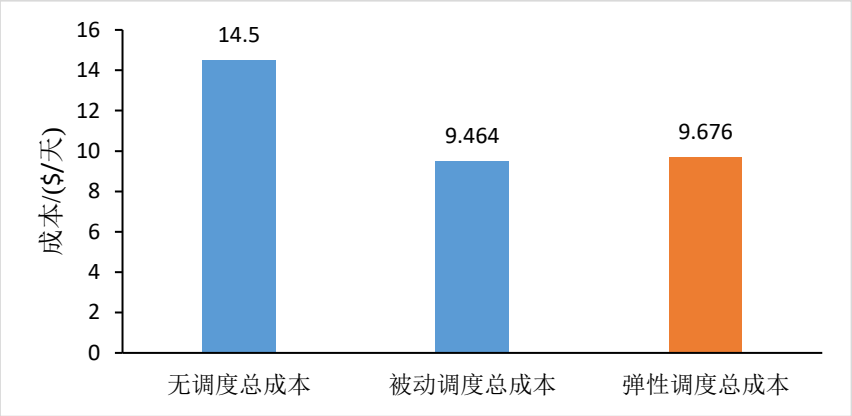

图 4.8　三种不同调度策略下的成本开销

从图 4.8 中可以看到，弹性调度总成本略高于被动调度总成本，但弹性调度不会影响键值存储系统的性能。相比于无调度策略，弹性调度策略可以节省 33.27%的成本开销。

（2）点餐服务应用负载场景下的测试

使用 db_bench 测试工具来模拟点餐服务应用的负载场景，对云服务器中的 RocksDB 进行压力测试，当 RocksDB 的负载请求变化处于一个稳定的范围后，开启 IESS，其中智能预测模块需在线上进行动态训练，以确保负载预测的准确度。

在相同负载场景下，采取无调度、被动调度和弹性调度这三种调度策略来对云服

务器的存储资源进行调度，RocksDB 在这三种调度策略下 QPS 的变化如图 4.9 所示：

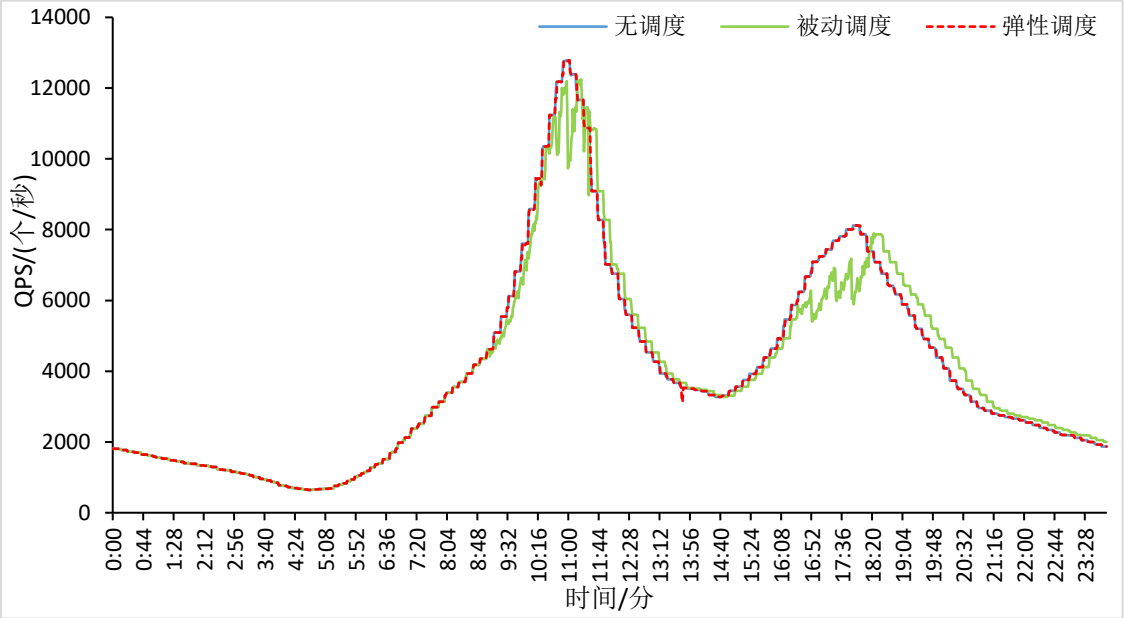

图 4.9　三种不同调度策略下 QPS 变化图

从图 4.9 中可以看到，表示无调度 QPS 的蓝色曲线与表示弹性调度 QPS 的红色虚线基本上重合在一起，而表示被动调度 QPS 的绿色曲线则有一定的滞后性。

QPS 越高表示性能越好，三种调度策略下 RocksDB 的平均 QPS 如图 4.10 所示，可以看到，弹性调度的平均 QPS 略低于无调度的平均 QPS，说明弹性调度的性能与无调度的性能很接近，而被动调度的平均 QPS 最低，说明被动调度的性能较差。

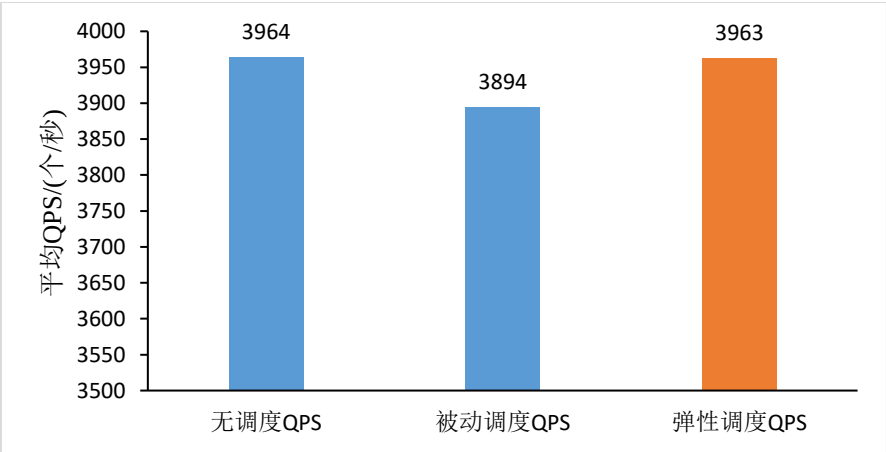

图 4.10　三种不同调度策略下平均 QPS

RocksDB 会根据 IESS 生成的调度指令对存储资源进行弹性调度，磁盘 IOPS 变

化如图 4.11 所示：

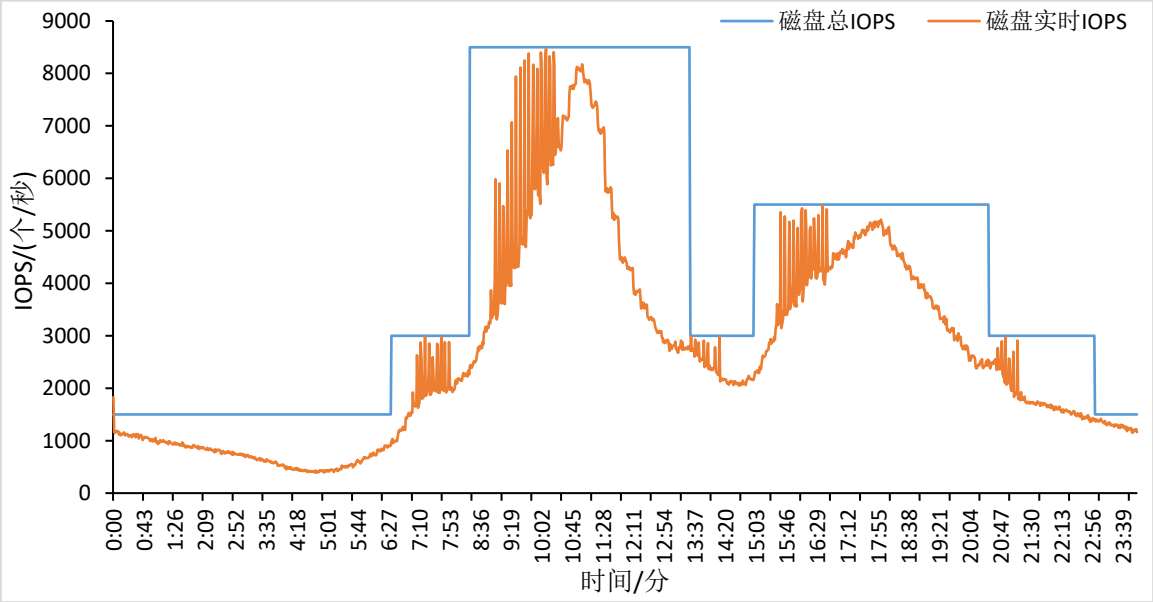

图 4.11　磁盘 IOPS 变化图

在图 4.11 中，蓝色曲线表示磁盘总 IOPS，橙色曲线表示磁盘实时 IOPS，可以看到，在前台负载增加时，RocksDB 会提前扩充存储资源，从而增加磁盘总 IOPS，而数据迁移的负载会使磁盘实时 IOPS 产生不规则波动，在前台负载下降时，RocksDB 会缩减存储资源，从而减少磁盘总 IOPS，由于不需要再进行数据迁移，因而磁盘实时 IOPS 不会产生不规则波动。

三种调度策略下，一天所需的存储资源如表 4.6 所示：

表 4.6　三种调度策略下一天所需的存储资源

调度策略	一天所需的存储资源
无调度	100GB 8500IOPS io2 24h
被动调度	100GB 1500IOPS io2 24h 50GB 1500IOPS io2 17h 50GB 2500IOPS io2 5h 50GB 3000IOPS io2 5h
弹性调度	100GB 1500IOPS io2 24h 50GB 1500IOPS io2 17h30min 50GB 2500IOPS io2 5h30min 50GB 3000IOPS io2 5h30min

云服务器中的存储资源价格为每 24 小时每 100GB 的费用为 0.4167 美元，每 24 小时每 1000IOPS 的费用为 2.1667 美元，由此可以计算表 4.6 中所需的存储资源的成本开销，如图 4.12 所示：

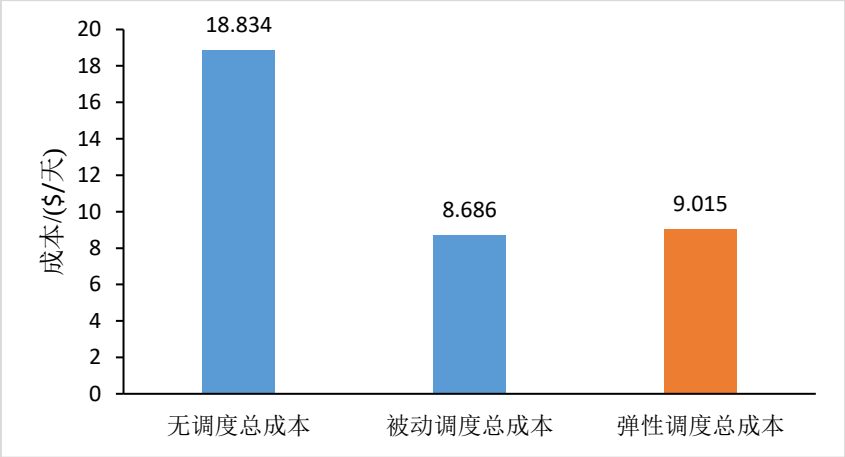

图 4.12　三种不同调度策略下的成本开销

从图 4.12 中可以看到，弹性调度总成本略高于被动调度总成本，但弹性调度不会影响键值存储系统的性能。相比于无调度策略，弹性调度策略可以节省 52.13%的成本开销。

4.3.3　测试小结

使用 db_bench 测试工具来模拟出行服务应用和点餐服务应用这两种负载场景对云服务器中的 RocksDB 进行压力测试，从图 4.5 和图 4.9 中可以看到，在基于负载预测的弹性调度策略下，RocksDB 的 QPS 变化曲线与无调度策略下的 QPS 变化曲线基本重合，而被动调度策略下的 QPS 变化曲线存在一定的滞后性。这说明被动调度策略会影响键值存储系统的性能而导致 QPS 下降，而基于负载预测的弹性调度策略会提前进行数据迁移操作，有效降低了对键值存储系统性能的影响。从图 4.8 和图 4.12 中可以看到，弹性调度总成本略高于被动调度总成本，两者都远低于无调度策略的总成本。相比于无调度策略，基于负载预测的 IESS 弹性调度策略在两种负载场景下可以节省 33.27%和 52.13%的成本开销。

4.4 本章小结

本章主要对实现的 IESS 进行完整的测试和分析，测试分为两部分：负载预测模型测试和键值存储系统智能弹性调度测试。首先，在线下环境中采用典型的互联网应用数据集进行预测分析，验证标准 RNN 模型、LSTM 模型、GRU 模型、DRNN 模型和 BRNN 模型在负载预测方面的准确度；其次，在线上环境中使用 db_bench 测试工具模拟互联网应用负载场景来进行压力测试，通过比较三种不同调度策略下 RocksDB 的 QPS 和成本开销来评估弹性调度策略的效果。通过实验得出，DRNN 模型中的 DRNN_LSTM 模型在负载预测方面的有着更好的表现，对于出行服务应用和点餐服务应用两种数据集，误差百分比与 R2_Score 分别达到了 5.09%、98.29%和 7.20%、97.74%，且基于负载预测的键值存储系统弹性调度能够在几乎不影响键值存储系统性能的情况下以较少的存储资源满足上层应用的运行需要，相比于无调度策略，IESS 弹性调度策略在两种负载场景下可以节省 33.27%和 52.13%的成本开销。

5 总结与展望

5.1 全文总结

信息技术的高速发展带动了互联网应用的快速发展，为了给用户提供优质的服务，同时降低生产成本，越来越多的企业在云上部署应用程序，并使用键值存储系统来存储应用数据。用户的行为习惯决定了互联网应用负载具有规律波动性，即在少数特定时间段处于高负载，而在其余时间段处于低负载。如果能够在高负载时扩充存储资源，在低负载时缩减存储资源，就能够极大减少企业的运营成本。但现有的调度策略大多数是被动调度，具有明显的滞后性，扩充存储资源后进行数据迁移产生的迁移负载会与前台负载竞争而导致键值存储系统性能下降。

针对上述问题，本文提出一种基于负载预测的键值存储智能弹性调度系统（IESS），IESS 可以收集云服务器中键值存储系统的实时负载请求，并利用神经网络模型来预测未来一段时间内的负载请求变化情况，生成调度指令并发送给键值存储系统。通过该调度指令，键值存储系统能够避开前台负载高峰，在低负载时刻提前进行调度操作，从而降低数据迁移给性能带来的影响。具体的研究工作如下：

（1）对真实互联网应用负载变化场景、存储资源调度场景的分析，指出互联网应用负载变化具有规律波动性以及被动调度会影响键值存储系统的性能，并因此提出要设计和实现 IESS，来实现键值存储系统对存储资源的弹性调度。

（2）采用多种循环神经网络对负载数据进行建模，根据测试结果选择负载预测效果最好的 RNN 模型来实现 IESS 的智能预测模块，并通过在线上动态训练来保证负载的准确度；基于预测负载信息，IESS 的智能调度模块能够在低负载时刻将热度值高的 SSTable 数据文件迁移到新扩充的存储资源中，避免数据迁移产生的读写负载与前台负载高峰产生竞争，并在负载请求下降时，对存储资源进行缩减，实现了键值存储系统对存储资源的弹性调度。

（3）对实现后的 IESS 进行大量测试，测试主要可以分成两部分：负载预测模型测试和键值存储系统智能弹性调度测试。测试表明：在预测方面，DRNN_LSTM 模型的预测效果最好，对于出行服务应用和点餐服务应用两种数据集，误差百分比与

R2_Score 分别达到了 5.09%、98.29%和 7.20%、97.74%，说明负载预测的结果是相当可靠的；在调度方面，基于负载预测的键值存储系统弹性调度能够在几乎不影响键值存储系统性能的情况下以较少的存储资源满足上层应用的运行需要，相比于无调度策略，IESS 弹性调度策略在两种负载场景下可以节省 33.27%和 52.13%的成本开销。

5.2 研究展望

通过 IESS 对实时负载请求数据进行预测并生成调度指令，能够实现键值存储系统对存储资源的弹性调度，在几乎不影响键值存储系统性能的情况下以较少的存储资源满足上层应用的运行需要。但是该系统仍然存在一些不足之处和可以改进的空间，具体如下：

（1）虽然使用了多个 RNN 模型来进行负载预测，并比较了它们的优劣，但是在训练时不同神经网络模型采用的参数基本上是一样的，可能没有完全发挥出每种模型的优势。对于同一个神经网络模型，可以设置多组不同的参数组合来进行对比实验，找到可以发挥每种模型优势的最佳参数组合。

（2）目前仅是对云服务器中键值存储系统的所有负载请求进行预测，后面可以考虑将负载数据细化到单个应用上，这样可以带来更好的预测效果。

（3）键值存储系统的弹性调度依赖于 IESS 的负载预测，因此负载预测准确度的高低决定了键值存储系统弹性调度的好坏。在系统生成调度指令时，可以将预测模型的实时准确度作为参考依据，只有当实时准确度达到一定的阈值后，系统才会将该调度指令发送给键值存储系统来执行调度操作，否则生成的调度指令只会在系统中进行显示而不会对云服务器中的键值存储系统产生任何影响。

参考文献

[1] Benoit Dageville, Thierry Cruanes, Marcin Zukowski, Vadim Antonov, Artin Avanes, Jon Bock, et al. The Snowflake Elastic Data Warehouse. In: Proceedings of the 2016 International Conference on Management of Data (SIGMOD/PODS 2016), San Francisco, USA, June 26-July 1, 2016: 215-226

[2] Alexandre Verbitski, Anurag Gupta, Debanjan Saha, Murali Brahmadesam, Kamal Gupta, Raman Mittal, et al. Amazon Aurora: Design Considerations for High Throughput Cloud-Native Relational Databases. In: Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD/PODS 2017), Chicago, USA, May 15-19, 2017: 1041-1052

[3] Alexandre Verbitski, Anurag Gupta, Debanjan Saha, James Corey, Kamal Gupta, Murali Brahmadesam, et al. Amazon Aurora: On Avoiding Distributed Consensus for I/Os, Commits, and Membership Changes. In: Proceedings of the 2018 International Conference on Management of Data (SIGMOD/PODS 2018), Houston, USA, June 10-15, 2018: 789-796

[4] Qi Lin. Analysis and Discussion of Component in the Ecosphere of Hadoop from HuaWei'FusionInsight. In: 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE 2021), Nanchang, China, March 26-28, 2021: 713-717

[5] Wei Cao, Yingqiang Zhang, Xinjun Yang, Feifei Li, Sheng Wang, Qingda Hu, et al. PolarDB Serverless: A Cloud Native Database for Disaggregated Data Centers. In: Proceedings of the 2021 International Conference on Management of Data (SIGMOD/PODS 2021), Xi'an, China, June 20-25, 2021: 2477-2489

[6] Wei Lu, Zhanhao Zhao, Xiaoyu Wang, Haixiang Li, Zhenmiao Zhang, Zhiyu Shui, et al. A Lightweight and Efficient Temporal Database Management System in TDSQL. Proceedings of the VLDB Endowment, 2019, 12(12): 2035-2046

[7] Andreas Meier, Michael Kaufmann. SQL & NoSQL Databases. Wiesbaden: Springer Vieweg, 2019: 201-218

[8] Siying Dong, Mark Callaghan, Leonidas Galanis, Dhruba Borthakur, Tony Savor,

Michael Stumm. Optimizing Space Amplification in RocksDB. In: the 8th Biennial Conference on Innovative Data Systems Research (CIDR 2017), Chaminade, USA, January 8-11, 2017: 3-3

[9] Jinhong Li, Qiuping Wang, Patrick P. C. Lee, Chao Shi. An In-Depth Analysis of Cloud Block Storage Workloads in Large-Scale Production. In: 2020 IEEE International Symposium on Workload Characterization (IISWC 2020), Beijing, China, October 27-30, 2020: 37-47

[10] Shuohan Chen, Yuhong Liang, Mingchang Yang. KVSTL: An Application Support to LSM-Tree Based Key-Value Store via Shingled Translation Layer Data Management. IEEE Transactions on Computers (Early Access), 2021: 1-1

[11] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, et al. Bigtable: A Distributed Storage System for Structured Data. ACM Transactions on Computer Systems, 2008, 26(2): 1-26

[12] Mehul N. Vora. Hadoop-HBase for Large-Scale Data. In: Proceedings of 2011 International Conference on Computer Science and Network Technology (ICCSNT 2011), Harbin, China, December 24-26, 2011: 601-605

[13] Avinash Lakshman, Prashant Malik. Cassandra: A Decentralized Structured Storage System. ACM SIGOPS Operating Systems Review, 2010, 44(2): 35-40

[14] Rebecca Taft, Irfan Sharif, Andrei Matei, Nathan VanBenschoten, Jordan Lewis, Tobias Grieger, et al. CockroachDB: The Resilient Geo-Distributed SQL Database. In: Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD/PODS 2020), Portland, USA, June 14-19, 2020: 1493-1509

[15] Dongxu Huang, Qi Liu, Qiu Cui, Zhuhe Fang, Xiaoyu Ma, Fei Xu, et al. TiDB: A Raft-Based HTAP Database. Proceedings of the VLDB Endowment, 2020, 13(12): 3072-3084

[16] Mahendra P. Yadav, Nisha Pal, Dharmendar K. Yadav. Workload Prediction over Cloud Server Using Time Series Data. In: 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence 2021), Noida, India, January 28-29, 2021: 267-272

[17] In K. Kim, Wei Wang, Yanjun Qi, Marty Humphrey. CloudInsight: Utilizing a Council of Experts to Predict Future Cloud Application Workloads. In: 2018 IEEE 11th

International Conference on Cloud Computing (CLOUD 2018), San Francisco, USA, July 2-7, 2018: 41-48

[18] Zhichao Cao, Siying Dong, Sagar Vemuri, David H. C. Du. Characterizing, Modeling, and Benchmarking RocksDB Key-Value Workloads at Facebook. In: the 18th USENIX Conference on File and Storage Technologies (FAST 2020), Santa Clara, USA, February 24-27, 2020: 209-223

[19] Takaya Miyazawa, Hiroaki Harai, Yusuke Yokota, Yasushi Naruse. Sparse Regression Model to Predict a Server Load for Dynamic Adjustments of Server Resources. In: 2019 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN 2019), Paris, France, February 19-21, 2019: 249-256

[20] Haohan Sha, Majd Moujahed, Dahai Qi. Machine Learning-Based Cooling Load Prediction and Optimal Control for Mechanical Ventilative Cooling in High-Rise Buildings. Energy and Buildings, 2021, 242: 110980

[21] Shengwei Lv, Gang Liu, Xue Bai. Multifeature Pool Importance Fusion Based GBDT (MPIF-GBDT) for Short-Term Electricity Load Prediction. In: 2021 International Conference on Energy Engineering, New Energy Materials and Devices (NEMD 2021), Shenzhen, China, January 15-17, 2021: 012012

[22] 王萍，付晓聪，许海洋. 云计算中基于负载预测的虚拟资源调度策略. 青岛农业大学学报(自然科学版)，2020，37(01): 73-78

[23] 郑睿程，顾洁，金之俭，彭虹桥，蔡珑. 数据驱动与预测误差驱动融合的短期负荷预测输入变量选择方法研究. 中国电机工程学报，2020，40(2): 487-500

[24] Alberto Mozo, Bruno Ordozgoiti, Sandra Gómez-Canaval. Forecasting Short-Term Data Center Network Traffic Load with Convolutional Neural Networks. PLOS ONE, 2018, 13(2): e0191939

[25] Koné K. Désiré, Kouassi A. Francis, Konan H. Kouassi, Eya Dhib, Nabil Tabbane, Olivier Asseu. Fractional Rider Deep Long Short Term Memory Network for Workload Prediction-Based Distributed Resource Allocation Using Spark in Cloud Gaming. Engineering, 2021, 13(3): 135-157

[26]Kangji Li, Xianming Xie, Wenping Xue, Xu Chen. Hybrid Teaching-Learning Artificial Neural Network for City-Level Electrical Load Prediction. Science China-Information Sciences, 2020, 63(5): 1-3

[27]谢建群，刘怡俊，李生. 改进鲸鱼算法在云计算资源负载预测中的应用. 计算机工程与应用，2018，54(13): 73-77

[28]林珊，王红，齐林海，冯函宇，苏盈. 基于条件生成对抗网络的短期负荷预测. 电力系统自动化，2021，45(11): 52-60

[29]Trong D. Nguyen, Sang W. Lee. Optimizing MongoDB Using Multi-Streamed SSD. In: Proceedings of the 7th International Conference on Emerging Databases (EDB 2017), Busan, Korea, August 7-9, 2017: 1-13

[30]Jian Hu, Jin Hou. Big Data Query Optimization Based On MongoDB. Scientific Journal of Intelligent Systems Research, 2022, 4(1): 1-8

[31]Lu Qi, Guanghe Ge, Wanxi Deng, Yuqing Li, Gang Wang, Xiaoguang Liu, et al. NLOV: An Innovative Object-Oriented Storage System Based on BerkeleyDB. In: 2008 10th IEEE International Conference on High Performance Computing and Communications (HPCC 2008), Dalian, China, September 25-27, 2008: 958-963

[32]陈敬静，马明栋，王得玉. MongoDB 负载均衡算法优化研究. 计算机技术与发展，2020，30(3): 88-92

[33]Lei Wang, Guiqiang Ding, Yulong Zhao, Dingzeyu Wu, Chengrui He. Optimization of LevelDB by Separating Key and Value. In: 2017 18th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT 2017), Taipei, China, December 18-20, 2017: 421-428

[34]Ling Zhan, Kai Lu, Zhilong Cheng, Jiguang Wan. RangeKV: An Efficient Key-Value Store Based on Hybrid DRAM-NVM-SSD Storage Structure. IEEE Access, 2020, 8: 154518-154529

[35]Oana Balmau, Diego Didona, Rachid Guerraoui, Willy Zwaenepoel. TRIAD: Creating Synergies Between Memory, Disk and Log in Log Structured Key-Value Stores. In: 2017 USENIX Annual Technical Conference (USENIX ATC 2017), Santa Clara, USA, July 12-14, 2017: 363-375

[36] Oana Balmau, Florin Dinu, Willy Zwaenepoel. SILK: Preventing Latency Spikes in Log-Structured Merge Key-Value Stores. In: 2019 USENIX Annual Technical Conference (USENIX ATC 2019), Renton, USA, July 10-12, 2019: 753-766

[37] Pandian Raju, Rohan Kadekodi, Vijay Chidambaram, Ittai Abraham. PebblesDB: Building Key-Value Stores Using Fragmented Log-Structured Merge Trees. In: Proceedings of the 26th Symposium on Operating Systems Principles (SOSP 2017), Shanghai, China, October 28-31, 2017: 497-514

[38] Philipp Hoenisch, Ingo Weber, Stefan Schulte, Liming Zhu, Alan Fekete. Four-Fold Auto-Scaling on a Contemporary Deployment Platform Using Docker Containers. In: the 13th International Conference on Service Oriented Computing (ICSOC 2015), Goa, India, November 16-19, 2015: 316-323

[39] Chuanqi Kan. DoCloud: An Elastic Cloud Platform for Web Applications Based on Docker. In: 2016 18th International Conference on Advanced Communication Technology (ICACT 2016), PyeongChang, Korea, January 31-February 3, 2016: 478-483

[40] Chenggang Wu, Vikram Sreekanti, Joseph M. Hellerstein. Autoscaling Tiered Cloud Storage in Anna. Proceedings of the VLDB Endowment, 2019, 12(6): 624-638

[41] Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning. Cambridge: MIT press, 2016: 368-375

[42] Jürgen Schmidhuber. Deep Learning in Neural Networks: An Overview. Neural networks, 2015, 61: 85-117

[43] Sepp Hochreiter, Jürgen Schmidhuber. Long Short-Term Memory. Neural computation, 1997, 9(8): 1735-1780

[44] Kyunghyun Cho, Bart V. Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, et al. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014), Doha, Qatar, October 25-29, 2014: 1724-1734

[45] Alex Graves, Abdel-rahman Mohamed, Geoffrey Hinton. Speech Recognition with Deep Recurrent Neural Networks. In: 2013 IEEE International Conference on

Acoustics, Speech and Signal Processing (ICASSP 2013), Vancouver, Canada, May 26-31, 2013: 6645-6649

[46] Alexander G. Parlos, Amir F. Atiya, K. Chong, Wei K. Tsai, Benito Fernandez. Recurrent Multilayer Perceptron for Nonlinear System Identification. In: IJCNN-91-Seattle International Joint Conference on Neural Networks (IJCNN 1991), Seattle, USA, July 8-12, 1991: 537-540

[47] Mike Schuster, Kuldip K. Paliwal. Bidirectional Recurrent Neural Networks. IEEE transactions on Signal Processing, 1997, 45(11): 2673-2681

[48] Alex Graves, Santiago Fernández, Jürgen Schmidhuber. Bidirectional LSTM Networks for Improved Phoneme Classification and Recognition. In: the 15th International Conference on Artificial Neural Networks (ICANN 2005), Warsaw, Poland, September 11-15, 2005: 799-804

[49] Nanyi Fei, Yizhao Gao, Zhiwu Lu, Tao Xiang. Z-Score Normalization, Hubness, and Few-Shot Learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV 2021), Montreal, Canada, October 11-17, 2021: 142-151

[50] Zhilu Zhang, Mert Sabuncu. Generalized Cross Entropy Loss for Training Deep Neural Networks with Noisy Labels. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems (NeurIPS 2018), Montreal, Canada, December 3-8, 2018: 8792-8802

[51] Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization. In: the 3rd International Conference on Learning Representations 2015 (ICLR 2015), San Diego, USA, May 7-9, 2015: 1-15