华南理工大学

South China University of Technology

硕士学位论文

基于心电动力学图和 Storm 的心肌缺血早期诊断平台的设计与实现

作 者 姓 名	阮润学
学 科 专 业	控制理论与控制工程
指 导 教 师	王聪
所 在 学 院	自动化科学与工程学院
论文提交日期	2017 年 4 月

Design and Implementation of Myocardial Ischemia Early Diagnosis Platform Based on Cardiodynamicsgram and Storm

A Dissertation Submitted for the Degree of Master

Candidate： Ruan Runxue

Supervisor： Prof. Wang Cong

South China University of Technology

Guangzhou, China

分类号：　　学校代号：10561

学　号：201520112611

华南理工大学硕士学位论文

基于心电动力学图和 Storm 的心肌缺血早期诊断平台的设计与实现

作者姓名：阮润学　　指导教师姓名、职称：王聪 教授

申请学位级别：工学硕士　　学科专业名称：控制理论与控制工程

研究方向：智能医疗系统

论文提交日期：　　年　月　日　　论文答辩日期：　　年　月　日

学位授予单位：华南理工大学　　学位授予日期：　　年　月　日

答辩委员会成员：

主席：_____

委员：_____

华南理工大学
学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名：　　日期：　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属华南理工大学。学校有权保存并向国家有关部门或机构送交论文的复印件和电子版，允许学位论文被查阅（除在保密期内的保密论文外）；学校可以公布学位论文的全部或部分内容，可以允许采用影印、缩印或其它复制手段保存、汇编学位论文。本人电子文档的内容和纸质论文的内容相一致。

本学位论文属于：

ϒ保密，在_____年解密后适用本授权书。

ϒ不保密,同意在校园网上发布,供校内师生和与学校有共享协议的单位浏览；同意将本人学位论文提交中国学术期刊(光盘版)电子杂志社全文出版和编入XNKI《中国知识资源总库》,传播学位论文的全部或部分内容。

(请在以上相应方框内打□□□)

作者签名：　　日期：

指导教师签名：　　日期

作者联系电话：　　电子邮箱：

联系地址(含邮编)：

摘 要

目前，国民的生活随着我国经济的飞速发展正发生着巨大的变化，人们生活节奏的加快使得心血管疾病发病率持续升高，心血管疾病已经严重危害人们的生命健康。心肌缺血是一种常见的心血管疾病，会使得心肌细胞缺氧从而导致心脏病变。临床上通常通过心电图检查对心肌缺血进行诊断，但是这种方法依赖于医务人员的临床经验，而且效率和准确率并不高。随着计算机技术的快速发展，借助计算机技术对心肌缺血进行诊断已经成为一种趋势。

确定学习理论可以对标准十二导联心电图中的 ST-T 段进行动力学建模，在学习训练之后可以得出心电动力学图（CDG）。由于健康人与心肌缺血患者的 CDG 有明显的不同，因此通过 CDG 可以准确地对心肌缺血进行早期诊断。如今，心电数据的规模越来越大，已有的单机诊断程序已经暴露出计算能力不足的缺点。此外，基于 Hadoop 平台的诊断程序虽然解决大规模数据批量处理的问题，但系统的实时性不足，而且缺乏对病人信息管理的功能。

针对上述的问题，结合目前流行的大数据处理以及云计算技术，本文实现了基于心电动力学图和 Storm 实时流计算框架的心肌缺血早期诊断平台。该平台采用 B/S 架构，一共分为计算层、存储层、业务层和表现层四个层次。计算层实现了基于确定学习的心肌缺血诊断程序，得益于 Storm 集群在实时计算方面的优势，它能够为用户提供实时可靠的计算服务。除此之外，平台的存储层为数据提供了可靠的持久性存储，业务层作为各个层次的枢纽实现了平台大部分的业务逻辑，表现层为用户提供了一个简洁美观的操作界面。通过 Web 浏览器，医务人员就可以使用平台的各种功能，实现对心电数据的心肌缺血早期诊断以及对病人相关数据的管理，从而显著提升了工作的效率。

关键词：心肌缺血；确定学习；诊断平台；CDG；Storm

Abstract

Nowadays, with the development of our country's economy, people' lives have ecountered a great change. The fast-paced life has led to the high incidence of cardiovascular diseases, which is becoming a big treat to our health. Myocardial ischemia is a kind of such diseases that can lead to hypoxia of myocardial cell and further to myocardial infarction. In clinical, ECG is the frequently-used method to diagnose the myocardial ischemia. However, the accuracy of this method depends on the clinical experience, which is result in lacking of efficiency. With the advances of computer technology, it trends to make use of the computer technology in the aided diagnosis of myocardial ischemia.

The deterministic learning theory can build a dynamics model to the ST-T segment of the standard 12-lead ECG, which is able to generate the Cardiodynamicsgram (CDG) after training. It is an obvious difference from the CDG of heathy people and patients. As a result, CDG is able to become the early diagnosis method of myocardial ischemia. In recent years, more and more data is obtained from the ECG instrument. In the face of such a massive data, the existing stand-alone program lacks of compute power to finish the task. The Hadoop-based diagnosis system solves the large-scale data processing problem, however, the system is not suitable for the real-time data processing and lacks of the ability of data management.

Regarding the issue mentioned above, this paper implements a myocardial ischemia early diagnosis platform based on CDG and Storm using the popular big data processing and cloud computing technology. The platform utilizes B/S structure that is divided into four layer, respectively computation layer, storage layer, business layer and presentation layer. The computation layer implements the deterministic-learning-based myocardial ischemia diagnosis algorithm. Thanks to the advantage of Storm cluster in real-time processing field, the computation layer provides real-time and reliable computation services to the users. Besides, the storage layer provides reliable persistent storage for the ECG data. The business layer is the hub of the platform, implementing a majority of system service logic. The presentation layer integrates a simple and beautiful interface for users. Through a web browser, medical personnel can use various functionalities of the platform in order to diagnose myocardial ischemia and manage patient information, which achieves a significant improvement of work efficiency.

Keywords: Myocardial ischemia; Deterministic learning; Diagnosis platform; CDG; Storm

目录

第一章 绪 论

1.1 研究背景

伴随着改革开放以来我国国民经济的飞速发展，人民的生活发生了巨大的变化，生活质量也得到了极大的提高。与此同时，快速的生活节奏导致了各种心血管病危险因素不断增多，如高血压、吸烟酗酒、肥胖、体力活动不足、不合理的膳食等。另外，环境的恶化在一定程度上也增加了心血管疾病的发病率，例如颗粒物（PM）大气污染是致病的一种危险因素，尤其是 PM2.5（细颗粒物）[1]。

相关统计表明，我国心血管疾病的患者呈现快速增长态势，目前约有患者 2.9 亿人，且每年约 300 万人死于心血管疾病，死亡率高居各类疾病死亡构成的首位，其中在农村的死亡构成为 44.60%，在城市的死亡构成为 42.51%，心血管疾病已经成为危害国民健康的“第一杀手”[1-2]。令人担忧的是，近年来随着社会经济的快速发展，各领域工作人员的工作强度不断提升，越来越多年轻人处于亚健康状态，心血管疾病也趋于年轻化。特别是在互联网行业，IT 工作人员的猝死新闻在近几年屡屡出现，这些发生在我们身边的事往往令人扼腕叹息。心血管疾病的预防以及早期诊断已经成为了降低心血管疾病致死率的重中之重。

在众多心血管疾病当中，心肌缺血是一种较为常见的心血管疾病。心肌缺血，是指心脏的血液灌注减少，导致心脏的供氧减少，心肌能量代谢不正常，不能支持心脏正常工作的一种病理状态。氧是心肌细胞活动不可或缺的物质，心脏完全依赖于血液把氧输送给细胞，一旦缺血，就会引发缺氧。心肌细胞的缺氧会导致心脏活动所需的能量不足，引发心绞痛、心功能下降、心律失常等危害。心肌缺血最常见的原因是冠状动脉粥样硬化，另外还有炎症（血管闭塞性脉管炎等）、痉挛、先天性畸形等多种病因。心肌缺血在临床上会有许多不同的症状出现，如胸闷、心悸、胸骨后和心前区胸痛或紧缩样疼痛。另外还有无症状心肌缺血，指患者有心肌缺血的客观指标，但是缺乏临床症状，从而也因此往往容易被人们忽视。近年来，大量的研究发现，大约有 25%~50%的急性猝死者在生前无心绞痛史。无症状心肌缺血正在医疗以及其他研究领域中受到越来越多的重视。对心肌缺血的早期诊断以及治疗能够有效地预防心肌梗塞，使得心脏保持健康良好的状态，对提高国民的健康水平有着重要的意义。

临床上，诊断心肌缺血的方法有很多，其中冠状动脉造影是最具代表性的一种，其准确率很高，但价格高昂，且在诊断的过程中对人体有一定的创伤和存在并发症的风险，

因此无法作为广泛应用的诊断手段。另一种常见的诊断手段就是心电图诊断，心电图反应心脏活动的过程，具有丰富的病理信息，对于各种心肌梗死、心律失常、心肌缺血的诊断具有一定的意义。作为最普遍的一种诊断方法，心电图具有无创伤、便宜和安全等优点。但是心电图具有复杂多样的缺点，同一种病理的心电图可能有较大的差异，因此要求诊断医生具备丰富的临床经验和扎实的知识理论基础。然而在短时间内观察大量心电图后，医生容易出现疲劳、误诊等状况。借助计算机和信号处理技术对心电信号进行分析能够有效地减轻医生在分析心电图上的工作量，使得诊断工作更具效率与正确性。

传统的心电图诊断过程是患者到医院采集心电图数据，然后经医生诊断后将结果反馈给患者。近几年来，各种载有心电信号诊断算法的家用心电图机不断涌现，用户可以随时了解自身心脏的健康情况而省去大量医院排队挂号的时间。但是受限于硬件的计算能力，普通的家用心电图机只能进行简单的波形特征判断，无法做到对心电数据进行更深一层的分析，因此不能充分挖掘心电信号中的隐藏信息。随着互联网和云计算技术的进步，远程医疗得到了快速的发展。在心电图诊断方面，用户可以将本地测得的心电信号通过互联网传送到远端的云平台上，借助云平台强大的计算能力，充分挖掘心电信号上面的隐藏信息，从而提高诊断的实时性和准确性。

1.2 国内外研究发展现状

心电图（Electrocardiography, ECG）是通过胸腔皮肤上的电极以时间为单位记录心脏的电生理活动的一种技术，自 1903 年被 Einthoven 发明以来，被广泛应用于心血管病和心律失常的诊断，为预防与治疗心血管疾病作出了巨大的贡献[3]。早期的心电图的诊断分析完全依靠临床经验丰富的医生来完成，这样不仅会消耗大量时间，而且主观的诊断在某些情况下并不可靠。二十世纪五十年代以来，计算机技术开始应用于心电信号的分析与诊断过程，医生也得以从分析心电图的繁琐工作中解脱出来，从而能够专注于个别异常数据的分析[4]。

心电信号的分析过程大致可以分为预处理、特征点定位、特征提取与分析[5]。心电信号的预处理主要用于抑制信号中的噪声，使得信号曲线平滑，特征点突出。对心电信号进行预处理通常采用滤波的方法，常用的滤波方法有经典数字滤波器法、自适应滤波器法、神经网络法、数学形态法和小波变换法等[6]。通常，心电信号由以下几种特征波形组成：P 波、PR 间期、QRS 波群、ST 段、T 波以及 QT 间期。每一个波形在其形态、幅值和间期上都有特定的取值范围，因此这些特征波形可以作为心电信号分析与诊断的

重要依据。常用的特征波形定位方法有滤波器法、差分法、神经网络法、支持向量机法等[7-10]。心电信号分析的一个重要步骤就是对信号进行特征提取与分析，该过程可以在信号的时域中进行，也可以在变换域中进行。在众多变换方法中，傅里叶变换，小波变换，希尔伯特变换最为常用。心肌缺血在临床上的主要表现是心电图中的 ST-T 段发生异常变化，即 ST 段水平型或下斜型压低大于等于 0.05mV 和 T 波的低平与倒置[11]。研究心电图 ST-T 段的缺血性变化，能够为临床上的诊治提供依据，对预防和诊断心肌缺血具有重大的意义。目前，对心电图 ST-T 段的分析方法有许多种，例如经典的时域分析方法[12-15]、人工神经网络方法[16-18]和决策树方法[19]等。文献[20]提出一种基于确定学习理论的对心肌缺血进行早期诊断的科学方法，该方法利用 RBF 神经网络对心电信号中的 ST-T 段进行动力学建模，然后提取与心肌缺血相关的动态模式特征，并对特征进行更深层的分析诊断。

现有的心电数据智能分析技术虽然在预防与诊断心血管疾病上发挥了重大的作用，但随着心血管疾病患者数量的增加，一些难题也逐渐显现出来：

（1）缺乏实时的远程诊断。目前心电图的诊断大多都是经心电图机或心电监控设备采集心电数据后由医生进行诊断，无法实现实时的诊断。

（2）缺乏自助式的诊断。目前家用的心电诊断设备受限于设备的硬件水平，无法实现复杂的分析算法来对心电数据进行分析诊断。

（3）缺乏应对大数据的远程诊断系统。随着心血管疾病患者数量激增，待分析诊断的心电数据量也急剧增大，传统的心电诊断系统无法应对数量如此巨大的心电数据。

近几年来，互联网技术快速发展，大数据和云计算的技术的逐渐成熟为以上的难题提供了行之有效的解决方案。在互联网时代，社交网络、移动通信和电子商务等为人类社会带来了以“PB”为单位的巨量数据。根据 IDC（Internet Data Center）的估计，整个互联网的数据量在 2014 年已经达到了 4.4 ZB 左右，并预计在 2020 年将会翻 10 番[22]。面对海量的数据，传统的数据处理手段在数据获取、传输、存储和分析上都面临着巨大的挑战。分布式文件系统和 NoSQL 数据库被广泛应用于海量数据的存储。与传统数据库不同，NoSQL 数据库并不强调应用场景的统一，它有专门的 NoSQL 数据库对各种不同类型的应用的数据进行存储，这就更能适应不同的应用场合。谷歌公司的 GFS（Google File System，谷歌文件系统）是专门为存储数以百亿计的海量网页数据设计开发的分布式文件系统[23]。在 GFS 之上，谷歌公司还开发出实时表格系统 BigTable、在其上层的 MegaStore 以及跨数据中心的超级存储系统 Spanner[24-26]。除此之外，GFS 的开源实现

HDFS（Hadoop Distributed File System），Facebook 的 HayStack 都是比较著名的分布式文件系统。在分析处理海量数据的方面，传统的服务器显然存在处理能力不足的问题，为此，各大互联网公司分别推出了各自的大数据计算平台。大数据计算主要分为两种形式，批量计算模式和流式计算模式。批量计算具有高吞吐量、极强容错性、灵活的水平扩展性等优点。其中最为著名的当属谷歌公司设计的 MapReduce 计算范型[27]，随着 Hadoop 在业界中日渐流行，这种典型的批量处理计算范型已经在各大领域中获得了广泛的应用。与批量计算不同，流式计算强调的是实时性，很多应用场景对于计算的时效性有很高的要求，因此流式计算越来越受到各大领域的重视。流式计算常见的系统架构分为主从架构（Master-Slace）和 P2P 架构。主从架构中设有一个主控节点来管理全局节点，典型的例子是 Twitter 公司的 Storm 流计算框架；P2P 架构没有设置主控节点，因此在系统管理方面相对复杂，典型的例子是 Yahoo 公司的 S4 系统。不管是大规模批量计算系统还是流式计算系统，都是使用大规模的计算机集群来解决单台服务器无法解决的海量数据问题，它们在各自的应用场景中都发挥着重要的作用。

医疗领域会产生大量的医疗健康数据，据统计，普通的医疗机构每年会产生 1TB~2TB 的医疗数据，有些机构每年产生的数据量甚至达到 300TB~1PB[28]。大数据技术的日渐成熟，为充分挖掘海量医疗数据提供了有效的技术手段，为许多医学难题提供了新的解决途径。随着云计算技术的成熟，很多复杂的诊断算法得以实现，从而改变了一些疾病的诊断方法。借助计算机的分析，医生更能对疾病做出正确的诊断。文献[29]在 Hadoop 的基础上构建了一个医疗大数据挖掘平台，并对平台的基础层、平台层、功能层、业务层等的功能进行了详细的描述。文献[30]提出了基于大数据的智慧移动医疗信息系统结构，设计并实现了数据采集系统和临床护理信息系统。文献[31]在 Hadoop 平台上实现了基于确定学习的心肌缺血早期诊断算法，利用计算机集群有效地提高了算法的运行速度，实现了对大批量心电数据的快速诊断。然而，Hadoop 大数据计算框架属于批量计算，能够有效地提高系统的吞吐量，但实时性较差。在时效性要求较高的应用场景，例如对重症加强护理病房（ICU）病人的诊断，无法发挥出批量计算的优势。鉴于流式计算的高实时性特点，本文将在 Storm 框架上实现基于确定学习的心肌缺血早期诊断算法，并应用于心肌缺血早期辅助诊断平台，为心肌缺血诊断提供实时、高效的服务。另外，本文将在平台上实现相关的病人信息管理系统，提高平台的应用价值。

1.3 本文组织架构

本文将在 Storm 框架的基础上实现基于确定学习的心肌缺血早期诊断算法，并且开发相关的病人信息管理系统，从而搭建一个具备诊断与信息管理功能的心肌缺血早期辅助诊断平台，为医生诊断心肌缺血提供依据，同时也为患者提供一个自助的诊断平台。

本文共五章，具体内容安排如下：

第一章为绪论。首先，阐述了我国在心血管疾病上所面临的严峻形势以及心肌缺血病症的主要危害；然后，对目前国内外在心电信号分析上的研究现状与发展趋势进行了论述，并分析了大数据与云计算在医疗领域上的应用情况；最后，对本文的组织架构进行了简述。

第二章为心肌缺血诊断的相关知识以及确定学习理论。首先，介绍心电图的原理以及心肌缺血在心电图上的具体表现；然后，简单论述根据心电信号对心肌缺血的诊断方法；最后，重点介绍了确定学习理论，以及其在诊断心肌缺血上的应用。

第三章为心肌缺血辅助诊断平台的整体架构。首先对平台进行需求分析；然后设计平台的整体架构，包括计算层、存储层、业务层以及表现层；最后对平台的运行流程进行简单的介绍。

第四章详细介绍计算层的具体实现。首先对流式计算框架 Strom 进行介绍；然后介绍如何将原有的诊断程序移植到 Storm 框架上；最后，讨论将程序移植到 Storm 框架后的性能。

第五章平台的整体实现。首先，讨论数据存储层中存储结构的设计；然后，介绍后台业务层的逻辑实现，包括如何从存储层中获取数据、与表现层的交互方法等；最后，描述平台表现层的设计方法以及具体的实现细节。

最后一章是结论与展望。对本文所做的主要工作进行总结，并对平台的下一步改善提出展望。

第二章 确定学习理论与心肌缺血诊断

2.1 引言

心肌缺血是一种常见的心血管疾病，通常由冠状动脉硬化所引起。心电图是通过胸腔皮肤上的电极捕捉心脏电生理活动的一种常见的心血管疾病诊断手段，心肌缺血在心电图上的体现是 ST-T 段的异常改变。心肌缺血主要是通过影响心肌细胞的复极过程来体现在心电图上的变化，然而，在心肌缺血的早期，心电图上并不能明显体现这一过程，因此通过常规的心电图诊断并不能发现早期的心肌缺血。为了改进早期心肌缺血的诊断准确率，科研人员投入了大量的研究工作，提出了各种在心电图基础上的心肌缺血诊断方法。确定学习理论是建立在 RBF 神经网络基础上的一种在动态环境下的人工智能理论，利用该理论可以提取心电信号中隐藏的动态特征，并将其用于心肌缺血的早期诊断当中。本章将对心电图的基本原理以及心肌缺血在心电图上的具体表现进行介绍，然后简要介绍各种基于心电信号的心肌缺血诊断方法，最后将重点对确定学习理论的基本原理进行介绍，并讨论确定学习理论在早期心肌缺血诊断中的应用。

2.2 心肌缺血及其诊断方法

2.2.1 心脏传导系统与心电图原理

作为人体的“发动机”，心脏的主要功能的推动循环系统中的血液流动，从而为身体提供足够的氧气和养分，同时也带走新陈代谢的产物，维持人体的正常生命活动。心脏由心肌细胞构成，心肌细胞经过特殊的分化而具有起搏的功能，它们能使心脏产生周期性的搏动，从而产生周期性的电生理活动。心脏的传导系统主要由窦房结（Sinoatrial Node）、前结间束（Anterior Internodal Tract）、中结间束（Middle Internodal Tract）、后结间束（Posterior Internodal Tract）、房间结（Atrioventricular Node）、房间束（Bachmann’s Bundle）、左束支（Left Bundle Branch）、右束支（Right Bundle Branch）以及传导路径（Conduct Pathways）组成，如图 2-1 所示。

心脏正常的电生理活动始于窦房结，然后经结间束传导至房室结，接着顺着左、右束支传导从而兴奋心室。静息状态时，心肌细胞细胞膜外为正电荷，细胞膜内为负电荷，细胞膜内外存在着电位差，此时被称为极化状态。当心肌细胞受到刺激时，细胞膜改变对各种离子的通透性，电位差消失，此时被称为除极过程。除极过程不会持续太久，除极结束后心肌细胞细胞膜内外恢复原来的极化状态，这一过程被称为复极。这一系列有序的电信号传播引起一系列的电位变化，进而形成了心电图上相对应的波段。

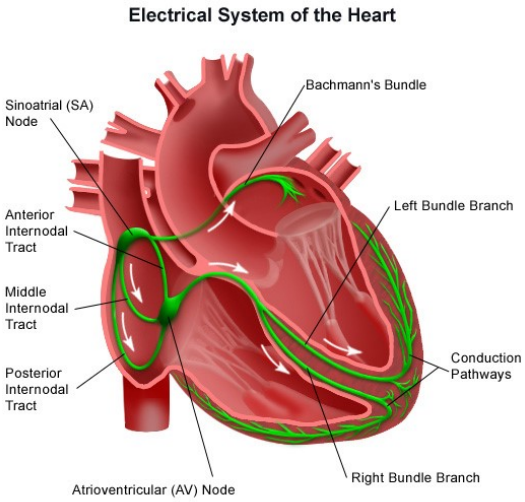

图 2-1 心脏传导系统

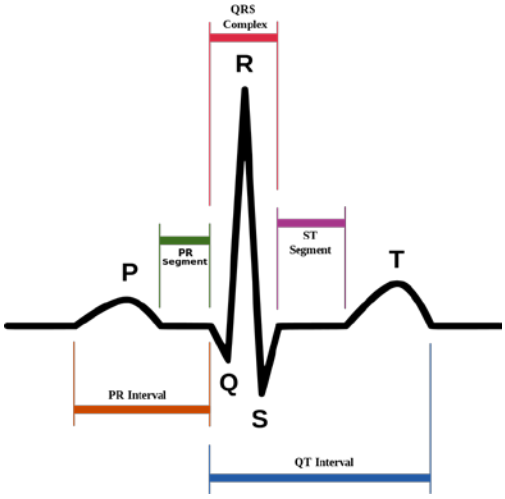

图 2-2 正常心电周期

心电图（Electrocardiography，ECG）是通过胸腔皮肤上的电极以时间为单位记录心脏的电生理活动的一种无创的诊断手段。如图 2-2 所示，一个正常的搏动周期主要由以下几个波段组成：P 波、PR 间期、QRS 波群、ST 段、T 波、QT 间期。这些波段反映了整个心脏的搏动节律，它们的具体意义如下所示[32]：

P 波：在正常的除极过程当中，除极从右心房延伸至左心房，形成 P 波。右心房的变化表现为 P 波的前半部分，左心房的变化表现为 P 波的后半部分。

PR 间期：从 P 波起始点到 QRS 波群起始点的一段时间，反映的是心房开始除极到心室开始除极的时间。

QRS 波群：反映心脏左、右心室的除极过程，振幅比 P 波高出很多。QRS 波群包含三个波段，按先后顺序依次是向下的 Q 波、向上的 R 波以及向下的 S 波。

ST 段：ST 段反映心室的缓慢复极过程，是从 QRS 波群结束点到 T 波起始点的一

条位于等位电势线上的水平线，是对心肌缺血临床诊断的重要指标。

T 波：反映心室的快速复极过程，是一个相对较低但占用时长较多的波段。心肌缺血可导致 T 波形态以及方向性的改变，例如 T 波低平或倒置。

QT 间期：QT 间期指从 QRS 波群的起始点到 T 波结束点之间的一段时间，代表心室从除极到复极的整个电位动作的过程。QT 间期与心率有关，心率越快，QT 间期越短；心率越慢，QT 间期越长。

2.2.2 心肌缺血生理机制及其诊断方法

心肌缺血指的是心脏的血液灌注减少，导致心脏的供氧减少，心肌能量代谢不正常，不能支持心脏正常工作的一种病理状态，是心血管疾病的常见病和多发病之一。长时间的血液供应不足会导致心肌缺血恶化为心肌细胞坏死，导致心肌梗死，从而很容易引起猝死。心肌缺血在生理机制上主要是由于心肌细胞的复极过程受到了影响，在心电图上的体现为 ST 段上的偏移以及 T 波在形态、方向上的改变[34]。

心电图是心血管疾病总要的辅助诊断手段，但是在诊断心肌缺血上存在着敏感性低、误诊率较高的缺点。近几十年以来，科研人员一直致力于寻找更为准确的心肌缺血诊断方法，提出了大量基于心电图的诊断心肌缺血的改进方法。

心电向量图（Vectorcardiography，VCG）诊断。心脏的电生理活动产生的电变化是立体的，根据这些电变化的动作方位以及大小的不同，按照心脏激动的顺序将从除极过程到复极过程所产生的三维心电向量的变化记录下来，即心电向量环，将向量环投影到三个相互垂直的平面上（横面、额面和侧面），就形成了 VCG[35]。心电向量环在各个平面的投影如图 2-3 所示。相比于普通的心电图，心电向量图更能准确地反映心脏的除极过程与复极过程，在鉴别诊断心肌缺血方面有着独特的优势[36]。

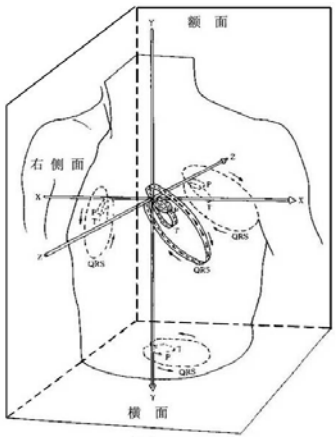

图 2-3 心电向量图

ST 段诊断。对心电图中的 ST 段在时域、频域或时频域上进行分析，是检测心肌缺血的常用手段。对 ST 段的分析可以分为形态上的分析以及大小上的分析。在形态上的分析方法有直线型及线性回归法、凹凸度及曲线类型法[37]、小波及形态法[38]；而在大小上的分析方法有趋势分析方法[39]。

T 波诊断。对 T 波进行心肌缺血的诊断方法主要是对 T 波电交替（TWA）的研究。T 波电交替指的是在常规心电图上 T 波的形态、幅度出现逐搏交替的变化[40]，与心肌缺血有着密切的联系。对 T 波电交替的检测分析方法主要有时域分析法和频域分析法。时域分析方法的代表是文献[40]提出的修正移动平均法（modified moving average method）；频域分析方法的代表是谱分析法（spectral method），该方法由 Smith 等人提出。

除此之外，常用的诊断心肌缺血的方法还有动态心电图、体表电位标测、运动平板试验以及心向量角检测法等等[42]。但是这些诊断方法在准确率上没能达到很高的水准，而且分析诊断过程相对复杂，较为依赖昂贵的软硬件，故很难广泛应用。文献[20]提出了一种基于确定学习的心肌缺血早期诊断算法，该算法对心电数据的 ST-T 段进行动态建模分析，是一种高准确率的创新诊断方法。

2.3 确定学习理论

确定学习理论[43,44]是一套建立在 RBF 神经网络上的学习机制，根据动力学系统以及自适应控制的方法，对动态环境下的未知知识进行学习、存储以及再利用。通过一个经过特殊设计的自适应神经网络控制器，确定学习理论能够在对周期性的轨迹进行控制过程当中，动态地学习未知的闭环系统的特性。

确定学习是针对动态系统的关于系统辨识和自适应控制的新的理论，它需要满足一下一些基本的条件[45]：

（1）需要使用局部径向基函数神经网络，即 RBF 神经网络（radial basis function neural network）。

（2）对于回归状态轨迹或周期轨迹需要满足部分持续激励（Persistence of Excitation，PE）条件。

（3）在回归状态轨迹或周期轨迹内利用 RBF 神经网络对未知的非线性闭环系统进行动态的局部准确逼近。

（4）对由动态学习所学得的知识以时不变且空间分布的形式表达，并存储为不变的神经网络权值。

相比于已有的神经网络学习方法，确定学习理论更能适应动态环境下对未知知识的学习，在许多应用中已经取得的较好的成效，如人体步态识别[46]、航空飞机轴流压气机旋转失速检测[47]、智能振动故障诊断[48]、心肌缺血辅助诊断[49]。

2.3.1 RBF 神经网络

径向基函数神经网络（Radial Basis Function Neural Networks），即 RBF 神经网络，是一种以径向基函数作为激活函数、用于局部逼近的人工神经网络。RBF 神经网络有许多用途，常用于函数的近似、模式分类、时间序列预测、自适应控制等领域。相对于 BP 人工神经网络而言，RBF 神经网络具有结构简单、逼近能力强、学习收敛等优点。通常，RBF 神经网络是一种具有单隐层的三层前向网络，如图 2-4 所示。第一层为输入层，由若干个感知神经元组成，将输入变量连接到隐含层的神经元中；第二层为隐含层，由若干个以径向基函数作为激活函数的神经元组成，将上一层的输出进行非线性变换，映射到隐含层的空间当中；第三层为输出层，对隐含层的输出进行加权线性组合并输出。

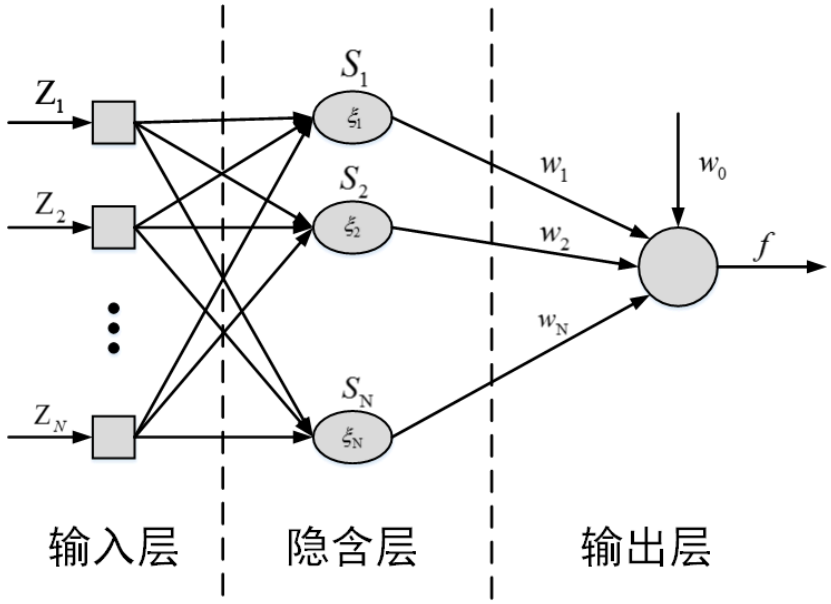

图 2-4 RBF 神经网络结构

RBF 神经网络属于线性参数模型，其在数学上的表达形式如公式(2-1)所示：

$$f_{nn}(Z)=\sum_{i=1}^{N} w_i s_i(Z)=W^T S(Z) \tag{2-1}$$

式中，$Z\in\Omega_Z\subset R^q$ 为输入变量；$W=[w_1,w_2,\cdots,w_N]^T\subset R^N$ 为权值向量；N 为神经网络节点数，取值为 $N>1$；$S(Z)=[s_1(\|Z-\xi_1\|),s_2(\|Z-\xi_2\|),\cdots,s_N(\|Z-\xi_N\|)]^T$，其中 $s_i(\bullet)$ 代表径向基函数，$\xi_i(i=1,2,\cdots,N)$ 代表状态空间上的中心点。高斯函数是常用的径向基函

数，其在数学上的表达式如公式（2-2）所示。

$$s_i(\| Z-\xi_i \|)=\exp\left[\frac{-(Z-\xi_i)^T(Z-\xi_i)}{\eta_i^2}\right], i=1,2,\cdots,N \tag{2-2}$$

式中，$\|\bullet\|$表示欧拉范数，$\xi_i=\left[\xi_{i1},\xi_{i2},\cdots,\xi_{iq}\right]^T$表示基函数的中心点，$\eta_i$表示基函数神经元之间在可接受区域的宽度，并且当$\| Z \|\to\infty$时，有$s_i(\| Z-\xi_i \|)\to 0$。

只要 RBF 神经网络中的神经元节点数 N 足够大，当节点的位置及节点之间的宽度选取恰当时，就能够在紧集$\Omega_Z\subset R^q$上以任意的精度对连续函数$f(Z):\Omega_Z\to R$进行逼近，其在数学上的表示如公式(2-3)所示：

$$f(Z)=W^{*T}S(Z)+\varepsilon(Z),\forall Z\in\Omega_Z \tag{2-3}$$

式中，W^*表示网络中理想的常数权值向量，$\varepsilon(Z)$代表逼近精度。

假设存在一个理想的权值向量W^*，对于任意$Z\in\Omega_Z\subset R^q$，使得$|\varepsilon(Z)|$最小，其在数学上的表示如公式(2-4)所示：

$$W^*\triangleq\arg\min_{W\in R^N}\left\{\sup_{Z\in\Omega_Z}|f(Z)-W^TS(Z)|\right\} \tag{2-4}$$

在 RBF 神经网络中，每个径向基函数只会对局部的输出产生影响，故对于紧集$\Omega_Z\subset R^q$上的任意有界轨迹$Z(t)(\forall t\ge 0)$，$f(Z)$能够通过轨迹附近的神经元对轨迹进行逼近，其在数学上的表示如公式(2-5)所示：

$$f(Z_p)=W_p^{*T}S_p(Z_p)+\varepsilon(Z_p) \tag{2-5}$$

式中，$\varepsilon(Z_p)$表示网络的逼近误差，$S_p(Z_p)=[s_{j1}(Z),s_{j2}(Z),\cdots,s_{jp}(Z)]^T\in R^{N_p}$是公式(2-1)中 S(Z)的子函数，$W_p^{*T}=\left[w_{j1}^*,w_{j2}^*,\cdots,w_{jp}^*\right]^T\in R^{N_p}$，其中$N_p<N$。

2.3.2 RBF 神经网络的持续激励条件

持续激励条件（persistent excitation condition），即 PE 条件，是自适应系统控制领域中的一个重要的概念。对于非线性系统而言，PE 条件能够确保系统参数收敛到期望值，然而这个条件一般比较难满足，其定义如下：

对于分段连续、一致有界的向量函数$S:[0,\infty)\to R^m$，若存在常数$\alpha_1>0,\alpha_2>0,T_0>0$，使得：

$$\alpha_1 I\le\int_{t_0}^{t_0+T_0}|S(\tau)^Tc|^2\,d\tau\le\alpha_2 I,\forall t\ge 0 \tag{2-6}$$

其中，单位矩阵 $I \in R^{m\times m}$，则该函数就满足了 PE 条件。

根据以上定义，要满足 PE 条件，半正定矩阵 $S(\tau)S(\tau)^T$ 在区间 $[t_0, t_0+T_0]$ 必须是正定的。

Kurdila 等人在文献[50]中提出了一种在连续和离散情况下都适用的持续激励条件的定义：

给定 $\mu \in [0,\infty)$ 为一个有限的 Borel 测量，对于一个分段连续、一致有界的向量函数 $S:[0,\infty)\to R^m$，若存在正常数 $\alpha_1>0, \alpha_2>0, T_0>0$ 使得：

$$\alpha_1 \| c \|^2 \le \int_{t_0}^{t_0+T_0} | S(\tau)^T c |^2 \, d\mu(\tau) \le \alpha_2 \| c \|^2 \tag{2-7}$$

对于任意常数向量 $c \in R^N$ 都成立，则成向量函数 S 满足 PE 条件。

此外，Kurdila 等人还在论文中提出以下定理：对于在时间区间 $[t_0, t_0+T_0]$ 的状态轨迹 $Z(t)(\forall t \ge 0)$，如果轨迹在每个神经元中心足够小的 δ 邻域里驻留时长 τ_0，其中 τ_0 与 t_0 无关，而 δ 满足 $0<\delta<h:=\frac{1}{2}\min_{i\ne j}\| \xi_i-\xi_j \|$，则称回归向量 $S_\xi(Z(t))$ 满足 PE 条件。然而，此定理对于 δ 所满足的条件过于 ，一个随机的周期轨迹在实际情况下可能无法对邻域内的所有节点进行访问，故不能运用于 RBF 神经网络中。为了改善这种情况，使得上述定理能够运用到 RBF 神经网络中去，确定学习理论放宽了对 δ 的限制：

对于任意回归轨迹或周期轨迹 Z(t)（周期为 T_0），若径向基函数子向量 $S_\xi(Z(t))$ 位于轨迹的 δ 邻域内，其中 δ 满足 $\delta \ge \sqrt{q}h = \frac{\sqrt{q}}{2}\min_{i\ne j}\| \xi_i-\xi_j \| > 0$，则称子向量 $S_\xi(Z(t))$ 满足 PE 条件。

放宽对 δ 的限制使得确定学习理论能够在动态的非线性系统中使用 RBF 神经网络对系统进行局部精确地逼近，从而实现对未知动态系统的局部精确建模。

2.3.3 离散系统下的确定学习

在利用确定学习理论对心肌缺血进行早期诊断时，所需要的是离散化的心电数据。本节将讨论在离散系统下确定学习理论的应用[51]。

对于如公式(2-8)所示的非线性离散系统：

$$Y(k) = F(Y(k-1),\cdots,Y(k-m); p) \tag{2-8}$$

式中，$Y=[y_1, y_2, \cdots, y_m] \in R^n$ 为可测量的系统状态，p 为系统常向量参数，而

$F(;p)=\left[f_1(;p),f_2(;p),\cdots,f_n(;p)\right]^T$ 表示连续未知的非线性光滑向量场。

假定 Y(k)一致有界（$Y(k)\in\Omega\subset R^n$，Ω 为紧集），对于回归或周期的系统状态轨迹，为了对系统未知的动态信息进行辨识，文献[51]构建了一个动态的 RBF 神经网络模型：

$$\hat{Y}(k)=-A(Y(k-1),\cdots,Z(k-1))+\hat{W}^T(k)S_k \tag{2-9}$$

其中：

$$Z(k)=-A(Y(k-1),\cdots,Z(k-1))+\hat{W}^T(k+1)S_k \tag{2-10}$$

式中，$\hat{Y}(k)=\left[\hat{y}_1(k),\hat{y}_2(k),\cdots,\hat{y}_n(k)\right]^T\in R^n$ 是对 Y(k) 的预先估计，而 $Z(k)=\left[z_1(k),z_2(k),\cdots,z_n(k)\right]^T\in R^n$ 则是对 Y(k) 的置后估计；A 为对角矩阵，即 $A=diag\left\{a_1,a_2,\cdots,a_n\right\}$，并且有 $|a_i|<1$；$\hat{w}(k)=\left[\hat{W}_1(k),\hat{W}_2(k),\cdots,\hat{W}_n(k)\right]$ 是 RBF 神经网络的估计权值向量；$S_k=S(Y(k-1),Y(k-2),\cdots,Y(k-m))$。

考虑系统(2-8)的一个子系统如公式(2-11)所示：

$$y(k)=f(Y(k-1),\cdots,Y(k-m);p) \tag{2-11}$$

设计一个 RBF 神经网络模型如公式(2-12)，(2-13)所示：

$$\hat{y}(k)=-\alpha(y(k-1),\cdots,z(k-1))+\hat{W}^T(k)S_k \tag{2-12}$$

$$z(k)=-\alpha(Y(k-1),\cdots,z(k-1))+\hat{W}^T(k+1)S_k \tag{2-13}$$

式中，$\|\alpha\|<1$，设模型的跟踪误差为 $e(k)=y(k-1)-z(k-1)$，则有：

$$e(k)=y(k-1)-z(k-1)=\alpha e(k-1)-\tilde{W}^T(k)S_{k-1}+\varepsilon \tag{2-14}$$

即：

$$e(k+1)=\alpha e(k)-\tilde{W}^T(k+1)S_k+\varepsilon \tag{2-15}$$

式中，$\tilde{w}=\hat{w}-w^*$，其中 w^* 是 RBF 神经网络的最优权值。

令 $v(k)=e(k+1)=y(k)-z(k)$，则有：

$$v(k)=y(k)-z(k)=\alpha e(k)-\tilde{W}^T(k+1)S_k \tag{2-16}$$

另外，令神经网络的权值更新率为：

$$\hat{W}(k+1)=\hat{W}(k)+\Gamma S_k v(k)-\sigma\hat{W}(k) \tag{2-17}$$

式中，$\sigma>0$ 为一个鲁棒常数。令 $y_k=y(k)$，$e_k=e(k)$，$z_k=z(k)$，$v_k=v(k)$，$\phi_k=-\tilde{W}^T(k+1)S(Y(k-1),Y(k-2),\cdots,Y(k-m))$。综合式(2-12)、(2-13)、(2-16)、(2-17)，

可以得到：

$$\begin{aligned} v_k &= y_k - z_k \\ &= y_k - \hat{y}_k + \hat{y}_k - z_k \\ &= y_k - \hat{y}_k - \left[\tilde{W}^T(k+1) - \tilde{W}^T(k)\right] S_k \\ &= y_k - \hat{y}_k + \sigma \hat{W}^T(k) S_k - S_k^T \Gamma S_k v_k \end{aligned} \tag{2-18}$$

由公式(2-18)可以得到：

$$v_k = \frac{y_k - \hat{y}_k + \sigma \hat{W}^T(k) S_k}{1 + S_k^T \Gamma S_k v_k} \tag{2-19}$$

将公式(2-19)替换公式(2-16)即可实现确定学习算法。文献[51]中的定理 2 证明，利用确定学习算法设计出来的神经网络模型确实能够对回归或周期的离散轨迹进行局部精确建模。

2.3.4 确定学习在心肌缺血早期诊断中的应用

心电图是描述心脏电活动的动态过程的一种记录手段，而且其波形是一类周期或回归轨迹。然而，目前基于心电图对心肌缺血进行诊断的大多方法都忽略了心电图所隐藏的动态信息，将心电信号作为静态模式进行研究，故诊断的准确率较低。确定学习是一种用于对动态模式进行准确建模和快速识别的算法，适用于检测非线性动力系统所产生的微小震荡故障，这与心肌缺血的诊断有着密切的联系。

基于确定学习的心肌缺血早期诊断是采用离散化后的确定学习算法，对常规的 12 导联心电图中的 ST-T 段进行建模分析，主要步骤如下：

1、对采集到的 12 导联心电数据进行预处理（包括导联体系转换和滤波）后进行 ST-T 段截取。

2、对截取后的 ST-T 段进行局部的准确建模。

3、从模型中提取心电信号的动力学信息。

4、通过绘制三维的心电信号动力学信息生成心电动力学图（cardiodynamicsgram，CDG）。由于健康人群的 CDG 形态较为规整，而心肌缺血患者的 CDG 形态较为散乱，因此根据 CDG 的形态可实现对心肌缺血的早期诊断。

CDG 是一种基于标准十二导联心电图的一种新型诊断方法。类似了传统的心电图，它具有无创性，但在心肌缺血的早期诊断上有着传统心电图无法比拟的准确性，因此可以广泛应用于心肌缺血的临床诊断中去。

2.4 本章小结

本章首先简单介绍了心脏的传导系统以及心电图的基本原理。然后对心肌缺血的生理机制进行了简要的描述，并对一些基于心电图的心肌缺血诊断手段进行了讨论。接着，重点介绍了与确定学习相关的基础理论，主要包括 RBF 神经网络及其持续激励的条件、离散系统下的确定学习算法。最后，讨论了确定学习理论在心肌缺血的早期诊断上的应用，通过对标准十二导联心电图的 ST-T 段进行动态建模，提取动力学信息，最后根据绘制的 CDG 的形态来实现对心肌缺血的早期诊断。

第三章 平台需求分析及架构设计

3.1 引言

前面章节对确定学习理论及其在心肌缺血早期诊断中的应用进行了详细的介绍。文献[31]在 Hadoop 平台上实现了基于确定学习的心肌缺血早期诊断算法，实现了对大规模心电数据进行批量分析的功能；文献[52]在 MATLAB 基础上开发了病人信息管理系统，实现了对病人的基本信息和心电数据的存储与管理。虽然 Hadoop 是大数据批量处理的主流技术，但是其实时响应的能力较差，很难满足高实时性的应用场景，例如 ICU 病房中的心肌缺血诊断。另外，基于 MATLAB 开发的信息管理系统是单机版，只支持单客户端的访问，而且随着数据量的剧增，单机的硬件水平无法满足海量数据的存储。本章将针对上述的几个问题给出一个有效可行的解决方案。

3.2 需求分析

如图 3-1 所示，本平台的功能主要有三个，分别是分析计算功能，数据存储功能和信息管理功能。其中，分析计算功能是利用确定学习算法，根据病人的心电数据进行心肌缺血的早期诊断；数据存储功能是将病人的基本信息、原始的心电数据以及诊断结果存储到服务器上；信息管理功能是对相关的信息、数据进行管理，以便进行更深入的研究。本节将对各个功能进行具体的需求分析。

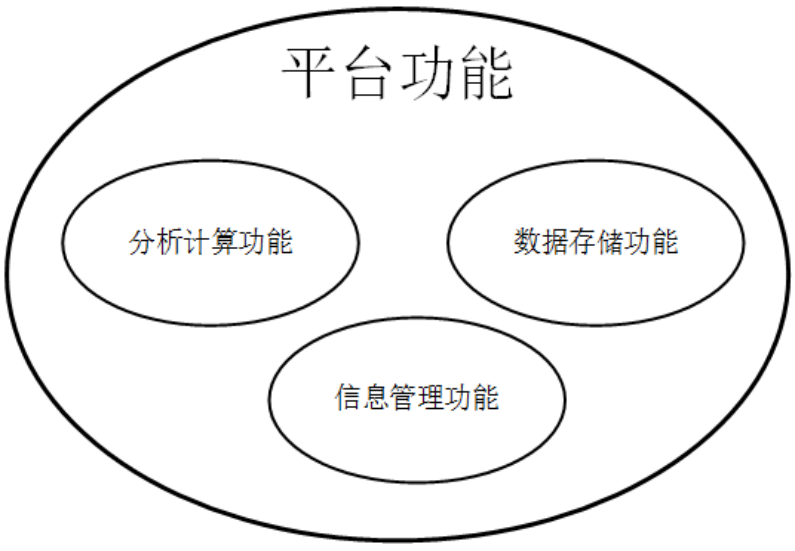

图 3-1 平台功能图

3.2.1 分析计算功能的需求分析

分析计算功能是本平台的核心功能，目的是将确定学习算法程序化，实现根据心电数据，对早期心肌缺血进行诊断。分析计算功能所要解决的问题如下所示：

1) 确定学习算法的计算机程序化。要将确定学习理论应用到心肌缺血的早期诊断中来，首先要解决的问题就是要将确定学习算法从数学上的表达转化为计算机能够理解的计算机程序。

2) 针对海量数据的处理能力。目前，中国的心血管疾病患者的数量巨大，要对海量的心电数据进行分析处理，无论在计算时间或是计算效率上，传统的数据处理技术已经无法满足需求。针对海量数据处理的大数据技术应该被应用到对心电数据的分析中来。

3) 高实时性。在某些应用场景，比如 ICU 重症监护病房，对诊断过程要求很高的实时性。

3.2.2 数据存储功能的需求分析

数据存储功能实现的是对病人相关数据的存储。利用本平台对病人进行心肌缺血的早期诊断时，会产生许多高研究价值的数据，例如病人的基本信息（年龄、性别等）、原始的心电数据、诊断结果等。数据存储功能要解决的问题如下所示：

1) 海量数据存储能力。随着时间的推移，平台所需存储的数据将不断增加，传统的单机存储已经无法满足对海量数据存储的要求。

2) 可扩展性。随着数据量的不断增大，当原有的服务器集群存储容量不足时，应该可以方便地对服务器集群进行扩展，以增大存储容量。

3) 容错性。在长时间运行的过程中，服务器集群中很难避免服务器能够一直稳定地工作着。当某台服务器宕机时，应该确保数据不会缺失。

3.2.3 信息管理功能的需求分析

信息管理功能为充分利用诊断过程产生的高研究价值数据提供了手段，使得更多隐藏在数据背后的知识被挖掘出来。信息管理功能要解决的问题如下所示：

1) 并发访问。已有的信息管理系统[51]只能允许一个客户端的访问，并不适应多用户使用的场景。作为一个面向大量用户的网络系统，本平台应该支持同一时间大量客户的并发访问。

2) 操作简单。信息管理系统面对的主要用户是医院的医务人员以及心肌缺血领域的研究人员，他们的计算机操作水平通常不是很熟练。因此，将信息管理的操作设计得简单方便，能够有效地提升用户的工作效率，使得他们免于长时间地学习操作该平台。

3) 界面友好。友好的界面与简单的操作一样，能够提高用户的工作效率，有利于

吸引更多的用户使用。

4) 负载均衡。大规模的并发访问使得必须对后台服务器进行负载均衡，否则可能存在某些服务器被频繁访问，导致效率低下，而且增加被频繁访问服务器宕机的可能性。

3.3 方案选取

3.3.1 分析计算方案选取

（1）算法实现方案

在将确定学习算法程序化上，有许多编程语言可供选择，常用的语言有 Java，C/C++，Matlab，Python 等。每种语言都有各自的优缺点，都有各自合适的应用场景。本人所在的团队已经在算法实现方面做了大量的工作量，目前，在基于确定学习的心肌缺血诊断算法上主要有 Matlab 版本[49]和 C++版本[42]。

Matlab 平台是工程师和科学家常用的研究工具，专门为解决工程以及科学问题做了优化。Matlab 在矩阵运算上面有着其他语言无法比拟的优势，而且内置的图形使得数据的可视化更加简单直观。众多的函数库和工具箱使得 Matlab 广泛应用在各个领域的研究上，包括信号处理、机器学习、计算机视觉、计算金融学、控制器设计等。随着大数据时代的到来，Matlab 也加入了对大型数据集进行运算分析的功能，并且可以扩展到集群和云上。然而 Matlab 的一些特点使得它并不适合搭建一个完整的应用后框架。首先，Matlab 不是一个开源的软件，在进行商业应用的开发时需要支付昂贵的费用，这也是 Matlab 没有在业界流行的主要原因。其次，Matlab 主要还是在 Windows 下运行，而且对硬件的要求非常高。在性能低下的 PC 上运行 Matlab 常常会因为 Windows 糟糕的资源管理机制，使得软件会占用大量的 CPU 资源。然后，Matlab 依赖于 Matlab 运行环境（Matlab Runtime Environment，MRE），而 MRE 需要占用较大的磁盘资源，而且初始化时间较长。最后，Matlab 与其他语言的交互性较差，虽然 Matlab 本身可以调用其他语言所编写的函数，Matlab 代码却很难被其他语言所使用。因此，本人所在团队所开发出来的基于 Matlab 的心肌缺血诊断系统只限制于团队使用，并不适用于大量用户的情况。

根据图 3-2 可以看出，在各类常用编程语言中，C 语言的运行速度最快，C++语言则紧随其后，另外两种常用的用于实现算法编程语言 Java 和 Python 则远不及前面两者。C 语言在程序运行上效率很高，但是由于 C 语言更接近底层，因此开发难度较大，而且在系统维护以及功能扩展方面也比较繁琐。C++语言是对 C 语言的进一步完善和扩充，

是一种面向对象的程序设计语言。而且 C++语言的适用性广，编译器、基础设施和库都

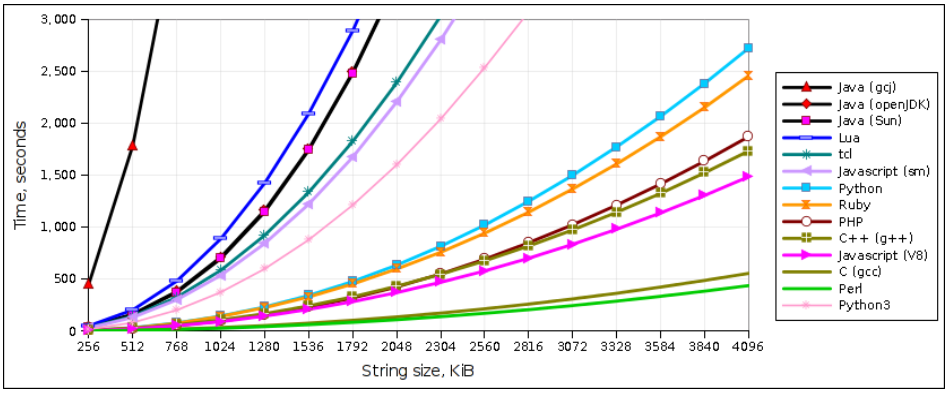

图 3-2 编程语言运行效率对比图

比较完善，很多框架都支持在 C++的基础上进行开发。

因此本平台将在基于确定学习的心肌缺血诊断算法的 C++版本的基础上进行设计和开发。

（2）计算框架方案

我们处于一个信息爆炸的时代，很多企业每小时就可以产生高达 10TB 的数据，传统的数据处理技术无法应对如此海量的数据，大数据、云计算技术应运而生。随着中国心血管病患者数量的不断增多，每天需要分析的心电数据剧增。为提高分析心电数据的的计算效率和计算时间，新兴的大数据和云计算技术应该被应用在心电数据的分析上来。目前主流的大数据计算技术主要分为两种，分别是批量计算以及流式计算。

批量计算是一种针对海量数据的具有高吞吐量、灵活的水平扩展能力、极强的容错性等优点的大数据计算技术。如图 3-3 所示，批量计算的一个特点就是数据必须要预先加载到系统中，后续的计算才能进行，因此批量计算是一种主动发起的计算。

图 3-3 批量计算模型

批量计算中最典型的范型当属 Google 于 2004 年发表的 MapReduce 计算范型，这种计算范型目前已经在众多领域中获得了广泛的应用。MapReduce 是一种构建在大规模普通 PC 之上的大规模计算框架，它实现了系统容错以及任务调度等分布式计算系统所必

须的功能，使得用户能专注于业务逻辑的实现。在众多 MapReduce 的开源实现中以 Hadoop 最为著名。Hadoop 是一个开源的针对海量数据的分布式计算平台，如图 3-4 所示，Hadoop 的核心设计思想就是将数据切片，并将其分配到成千上万的计算机集群中去，在每一台计算机中执行子任务，最后将结果汇总从而完成计算任务。

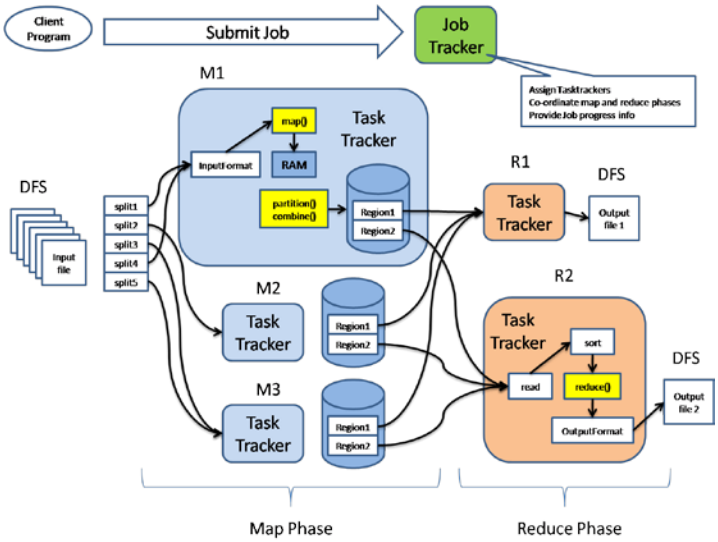

图 3-4 Hadoop 计算示意图

Hadoop 具有很多优点，有着强大的高效性、可靠性以及高扩展性。然而，Hadoop 也有明显的缺点：首先，Hadoop 的 MapReduce 擅长处理超大文件（通常是几百 M 到几百 T 这个级别的文件），对小文件的处理会大大降低性能的利用率；其次，Hadoop 的设计目的是大吞吐量，在实时性方面没有做一些优化，在对实时性要求很高的应用场景，Hadoop 并不合适[53]。

与批量计算模型不同，流式计算对计算的实时性要求更高。而且流式计算是一种常驻的计算服务，不同于批量计算的需要预先加载完全部数据才启动计算作业，流式计算一旦启动将处于等待状态，小批量的数据一旦到达消息队列，系统就会立即对数据进行计算并迅速将结果返回，从而具备了很高的实时性。流式计算的计算模型如图 3-5 所示。

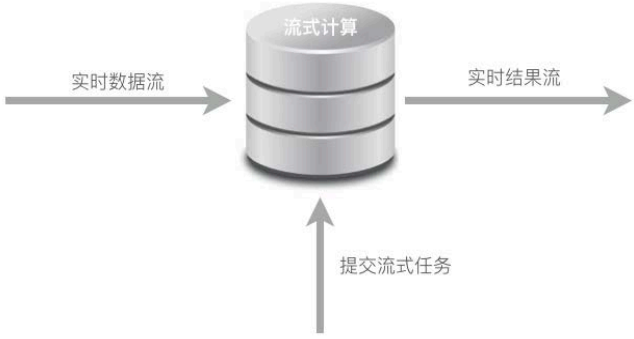

图 3-5 流式计算模型

目前主流的流式计算系统有很多，例如 Storm、S4、MillWheel、Samza 等，在这众多的系统当中，Storm 在各个方面的性能都比较突出。Storm 与 Hadoop 一样，同样是 Apache 旗下的一个顶级的项目，最初由 Twitter 公司开发并开源。Storm 有时候被称为实时处理领域的 Hadoop，其大大简化了海量数据流的处理机制，因而 Storm 之于实时处理领域就像 Hadoop 之于批量处理领域[54]。如图 3-6 所示，Storm 的设计思想是将“流”（数据输入流）与“栓”（处理与输出模块）结合起来从而构成一个有向无环图（Directed Acyclic Graph，简称 DAG）的拓扑结构，将拓扑结构运行在大规模集群之上，并由 Storm 本身根据拓扑的配置对任务进行分发。Storm 具有处理低延迟、系统容错性高、扩展能力强以及灵活的应用逻辑表达能力等优点，并且拥有丰富的流类型组合，足以应对任何类型的数据来源。

虽然 Hadoop 能够实现对海量数据的批量处理，但是其实时性不足，因此本平台将使用流式计算框架 Storm 作为基于确定学习的心肌缺血诊断程序的计算框架。

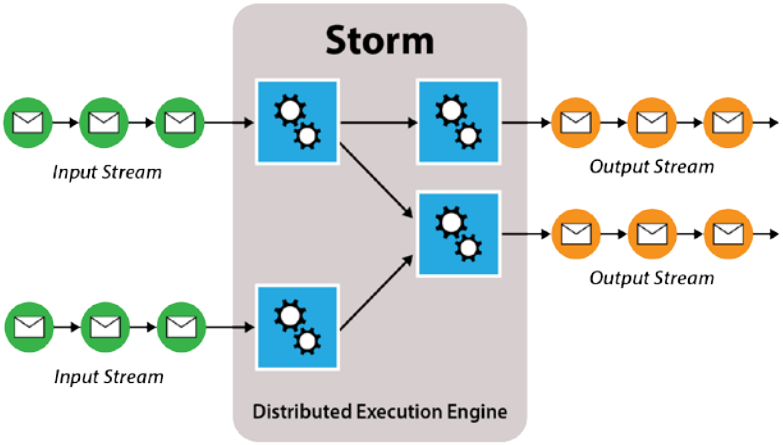

图 3-6 Storm 计算示意图

3.3.2 数据存储方案选取

不像 Hadoop 自身集成了一个分布式文件系统 HDFS，Storm 本身并没有集成一个数据存储系统，因此依靠外界的存储系统对计算结果进行持久化存储。另外，与诊断结果相关的数据如病人的基本信息、病历信息等也需要存储下来，以便对这些数据进行更深层次的挖掘研究。鉴于这些数据是结构化的数据，即心电数据、诊断结果、病人基本信息以及病历信息之间具有极强的关系，可以用一个二维表来表示，因此平台将选取传统的关系型数据库作为存储系统。常见的关系型数据库主要有 Oracle、Microsoft SQL Server 和 MySQL。

Oracle 关系型数据库是上述三者中出现最早的一个大型数据库，到目前为止仍然在

数据库市场上占有主要的份额。Oracle 具有严谨、稳定、高可用、高性能等优点，是上述三种关系型数据库中最安全的一个，因此被广泛地应用于金融、能源、通信、制造等各大行业的大型公司中。然而，由于 Oracle 的价格过于昂贵，功能过多而导致速度偏慢，因此目前并没有在互联网行业中流行开来。

Microsoft SQL Server 是微软公司开发，只面向 Windows 系统的关系型数据库。Microsoft SQL Server 面向的用户主要是中小型的企业，该数据库最大的优势在于其集成了微软公司的各类产品以及资源，为用户提供了强大的可视化界面。另外，高度集成的管理开发工具使得 Microsoft SQL Server 能够快速地构建商业智能，并使 Windows 系统在企业级应用中更加普及。但是 Microsoft SQL Server 有一个严重的缺点，就是过分依赖于 Windows 生态圈，与目前的主流服务器系统 Linux 并不兼容，因此在互联网企业中并不受欢迎。

不同于前面两者在使用时需要交付版权费，MySQL 是一个生于互联网，长于互联网的开源免费的小型关系型数据库。MySQL 主要应用于互联网的中小型网站的开发上，虽然体积不大， 但是其并发存取的能力并不比大型的关系型数据库差，再加上安装使用便捷，因此深受广大互联网公司的喜爱。MySQL 正在因为其性能高、可靠性好、成本低等优点逐渐成为目前最为流行的开源关系型数据库。MySQL 的最初开发者为瑞典的 MySQL AB 公司，后面被 Oracle 公司收购，成为 Oracle 公司旗下的产品。随着 Oracle 公司大幅上涨 MySQL 商业版的价格，导致 MySQL 存在闭源的潜在风险。因此，MySQL 的创始人麦克尔•维德纽斯主导开发了 MariaDB 数据库。MariaDB 是 MySQL 的一个分支，在应用程序的开发上与 MySQL 兼容，而且其由开源社区维护，使得其成为 MySQL 的一个代替品。

综合各方面的比较，本文所研究的平台将在 MariaDB 的基础上设计并开发系统的存储层。

随着数据量的不断增大，如果将所有的数据简单地存储在一个 MySQL 实例上，就无法对存储系统进行很好的扩展，这样迟早会遇到性能上的瓶颈，因此有必要对单个 MySQL 实例进行扩展。常用的扩展方式主要有垂直扩展、水平扩展、集群扩展以及向内扩展。

1) 垂直扩展。垂直扩展就是升级硬件，利用更高性能的及其来弥补大数据量带来的性能瓶颈。这种方案有许多好处，例如更加容易的开发与维护、更小的开销、无需考虑一致性的问题。然而，这种垂直扩展的策略只能够维持一段时间，如果数据量变得非

常庞大，很多问题就会显现出来。首先就是成本，高性能的硬件往往非常昂贵，然后就是单服务器在稳定性与容错性等方面存在着缺陷。因此，这种扩展策略一般被推荐。

2) 水平扩展。水平扩展可以分为复制、拆分以及数据分片。复制是最常用也是最简单的水平扩展方法，做法就是将数据通过复制分发到多个服务器上，并将备节点用于读查询。这是一种对于以读为主的应用非常有效的扩展方法，但存在数据重复缓存的问题，如果数据规模过大，这将会降低系统的效率。另一种水平扩展方法就是按功能拆分，也就是不同的 MySQL 节点执行不同的任务。而最通用和最成功的水平扩展方法是数据分片，该方法将数据分割成一小片，然后存储在不同的节点中。

3) 集群扩展。集群扩展的思想就是是单一逻辑数据库尽可能地存储更多的数据，处理更多的查询。集群扩展是将 MySQL 与集群或分布式数据库技术结合在一起，从而达到扩大存储容量的目的。常见的 MySQL 集群技术包括 MySQL Cluster、Clustrix、ScaleBase、GenieDB、Akiban 等，这些都具有自动扩展的能力。

4) 向内扩展。向内扩展就是对不断增多的数据进行归档以及清理，这通常会带来显著的成效，但是只能作为一个短期的策略，不能替代其他扩展方法。

在 MySQL 的扩展策略上，并不是一开始就要选择一个固定的扩展策略，通常的做法是，当应用的数据增大在一定的水平后，先从单个服务器架构向具有读备节点的架构转移，然后在进行数据分片或者按功能拆分。如果刚开始就进行数据分片，这将会是复杂而且代价昂贵的。就目前现有的心电数据数据量而言，本文所研究的平台将首先采取复制的水平扩展方案，一方面可以提高数据的容错性，另一方面可以采用读/写分离进行负载均衡。

3.3.3 信息管理方案选取

文献[52]所开发的信息管理系统只提供了单用户访问的解决方案，并未解决大规模并发访问的问题。目前，并发程序的实现架构主要分为两种，C/S 架构以及 B/S 架构。

C/S 是 Client/Server 的缩写，是一种客户端与服务器的架构，如图 3-7 所示，其通常拥有两层的结构。第一层是客户端，是运行于客户机系统之上的应用程序，结合了表示与业务逻辑的实现。第二层是数据库服务端，为客户端提供数据访问服务。C/S 架构的结构简单，而且点对点的模式使得系统更加安全。另外，客户端可以处理一些业务逻辑，从而在一定程度上减轻了服务端的压力。然而这种架

构也存在很多缺点，一方面是它面向的用户群是固定的，通常需要在用户 PC 上安装应用程序才能使用。另一方面是维护成本较高，每次系统的升级都会伴随着客户端的

一些更新，使得用户的使用更加繁琐。

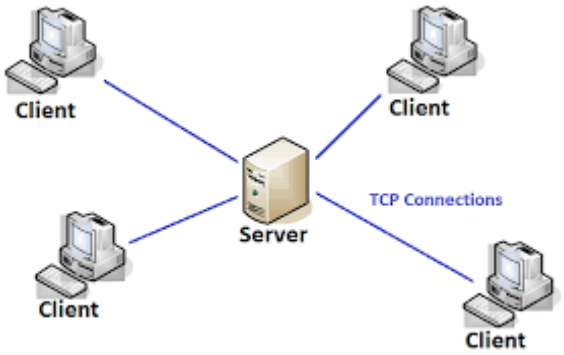

图 3-7 C/S 两层架构

文献[55]所设计的心电数据管理系统在传统的两层 C/S 架构上做了改善，在客户端与数据库服务器之间增加了一层中间件，使得整个系统的并发访问能力更强。该系统的架构图如图 3-8 所示。然而，客户端是基于 Matlab 开发的，严重依赖于 Matlab 开发环境，因此并不适合大规模的应用。

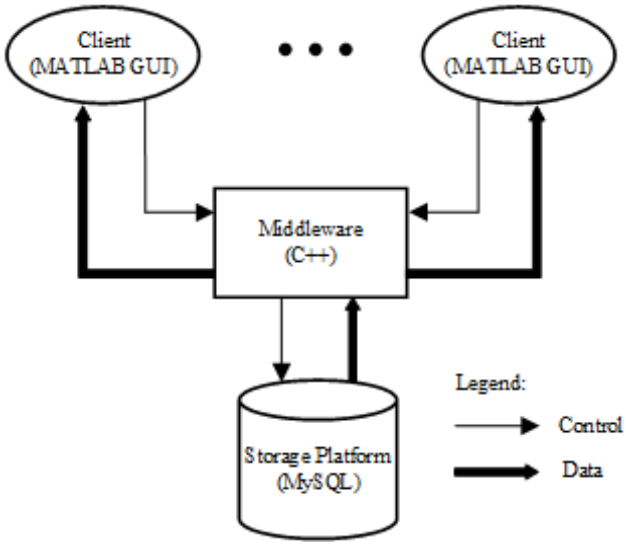

图 3-8 心电数据系统[53]架构图

B/S 是 Browser/Server 的缩写，是一种浏览器和服务器的架构，如图 3-9 所示，其通常具有三层结构。它是随着 Internet 和 WWW 的日渐流行在 C/S 架构之上的一种改进，可以说是一种三层的 C/S 架构。在 B/S 架构中，Web 浏览器将替代 C/S 架构中的客户端作为用户的操作界面，可以说，浏览器是一种轻量级的客户端。B/S 架构相对于 C/S 具有许多优点，例如无需安装客户端，几乎不用对客户端进行维护；能够有效地保护数据的访问权限，使得服务器数据库更加安全。

由于心肌缺血辅助诊断平台主要面向的是医务人员以及相关领域的研究人员，如果平台采用 C/S 架构会使得版本的升级更新花费成本巨大，而主要面向的用户由于缺乏软

件技能知识，因此使得维护成本更大。而如果采用 B/S 架构，用户只需通过常用的 Web

图 3-9 B/S 三层架构

浏览器就能使用平台的各种功能，不仅节省了用户大量的用于安装专门客户端的时间，平台的维护成本也大大减少。因此，本平台将采用 B/S 架构来实现。在图 3-9 所示的三层 B/S 架构当中，数据库系统（Database Server）相当于前一节所述的基于 MySQL 的数据存储层，而 Web 应用服务器（Web/Application Server）是实现系统主要业务逻辑的地方，Web 浏览器（Web Browser）则是用户与平台进行交互的操作界面。

Java 是用来实现 Web 应用服务器的最流行的的编程语言之一，Java 有着大量的库和框架，使得使用 Java 能够快速地搭建出一个可靠的 Web 应用服务器。Dropwizard 介于开发库与框架之间的 Java RESTful Web 框架，其目的是为 Web 应用所需的功能提供高性能、可靠的实现[56]。Dropwizard 将这些功能内置模块化，从而可以开发出小而精悍的应用程序，大大减少了开发和维护的难度。Dropwizard 使用具有高并发能力的 Jetty HTTP 库作为框架嵌入 HTTP 服务器，使得应用程序能够满足大规模的并发访问。Dropwizard 采用的是 RESTful 架构，一种目前最为流行的互联网软件架构。RESTful 架构结构清晰、易于理解、方便扩展，因此正在被越来越多的 Web 应用所采用。REST 是 Representational State Transfer 的缩写，即表现层状态转化。表现层其实就是资源（Resources）的表现层，RESTful 将网络上任一实体都看成是一个资源，可以用一个统一资源定位符（URI）来表示。而表现层（Representation）则是将这些资源表现出来的形式，例如可以用 txt 格式来表示文本，也可以使用 HTML 格式、JSON 格式、XML 格式来表示。URI 代表的仅仅是资源的实体而不是资源的形式。客户端要想与服务器进行交互，必须通过 HTTP 协议中的操作（GET、POST、PUT、DELETE）来时服务端发生状态转化（State Transfer）。由于这种状态转化是在表现层之上，因此叫做“表现层状态化”。

Web 浏览器是用户与系统进行交互的媒介，易用美观的操作界面可以给人带来良好的操作体验，使得用户有继续使用系统的动力。本文秉持着界面友好、操作简单的设计理念，对平台的前端界面进行设计开发。Boostrap 框架是一个基于 HTML、CSS、JavaScript 的前端框架，其具有简洁灵活的特点，使得 Web 前端的开发更为便捷。Boostrap 有大量设计良好的常用组件，开发者并不需要丰富的设计知识就能开发出美观的网页。

本文将采用 Dropwizard 作为 Web 应用服务器的架构，采用 Boostrap 作为平台前端界面的框架。另外，由于 Matlab 也支持 RESTful 风格的 API，只需对已有的 Matlab 客户端进行简单的更改便能兼容基于 Dropwizard 的 Web 应用。

3.4 架构设计

3.4.1 层次划分

有上一节的需求分析可知，本文所述平台一共包含了三方面的功能，分析计算功能、数据存储功能以及信息管理功能。综合这些功能的具体需求，本文将平台软件系统划分为四个层次：表现层、业务层、计算层以及存储层。其中各个层次的具体职责如下：

1) 表现层。表现层是平台与客户交互的层次，主要是负责心电数据的上传、病人基本信息以及病历信息的管理、诊断数据的可视化。

2) 业务层。业务层在整个平台中扮演者一个枢纽的角色，每个层次都存在与该层次的交互过程。首先，业务层接收表现层传来的心电数据和病人相关信息；然后，业务层将表现层传来的数据发送给存储层进行持久性存储，并将其中的心电数据传递给计算层进行进一步的分析诊断；接着，业务层会将计算层返回的诊断结果存到存储层中去；最后，业务层会根据表现层所发出的数据请求返回相应的数据。

3) 计算层。计算层是实现心肌缺血早期诊断算法的层次，该层次会将业务层发送过来的心电数据传送给 Storm 流式计算框架进行分析诊断，最后将诊断结果返回到业务层中去。

4) 存储层。存储层的主要职责是对各类数据进行持久性地存储，这些数据主要包括病人的基本信息、病历信息、原始心电数据以及诊断结果。

3.4.2 消息中间件

表现层与业务层之间的数据交互是通过 HTTP 协议实现的，业务层则通过 JDBI 与存储层进行数据传输，其中 JDBI 是一个轻量级的连接 SQL 的 Java 库。本文将采用一个消息中间件作为业务层与计算层之间的数据中转站。

Redis 是一个开源、高性能的 key-value 内存数据库，在存取的速度上，Redis 有着其他非内存数据库，如 MySQL，SQL Server 等，无法比拟的优势[57]。Redis 是一个存取速度极快的非关系型数据库，读的速度能够达到 110000 次/秒，写的速度能够达到 81000 次/秒。Redis 可以存储 key 与五种类型的 value 之间的映射，这五种类型的 value 分别是，字符串（String）、列表（List）、集合（Set）、哈希表（Hash）和有序集合（Sorted set）。这些复杂的数据类型都是在基本数据结构的基础上构建起来，并对开发人员完全透明。与其他内存数据库相比，Redis 主要有以下特点：（1）Redis 对数据类型的操作都是原子性的，并且支持几种操作合并后的原子性执行。（2）除了数据存储之外，Redis 还支持发布者/订阅者模式、通知、key 过期等特性。（3）Redis 可以将存储在内存中的数据持久化到磁盘中去。因此，在应用程序的开发中，Redis 一般被用作缓存系统或消息中间件。

一般而言，消息中间件有两种模式：一种是发布者/订阅者模式，一种是生产者/消费者模式。发布者/订阅者模式的运作过程是，发布者将消息放到消息队列里面，而多个监听队列的订阅者都能收到消息，而且消息应该都是一样的。生产者/消费者模式的运作过程是，生产者负责将消息放到消息队列中去，多个消费者同时监听这个消息队列，消息到达时就从队列中取走，因此每一个消息只能被最多一个消费者拥有。而这两种模式都可以利用 Redis 进行实现。

3.4.3 负载均衡

大量的用户访问会对 Web 应用服务器的性能提出非常高的要求，单台服务器往往不能承受如此大的网络负载。通过将多台服务器组成一个服务器集群，并通过负载均衡（Load Balancing）来对集群的请求任务进行分配，能够较好地解决了大规模 Web 并发访问所带来的问题[58]。

如图 3-10 所示，负载均衡的原理是客户端向负载均衡服务器发送请求，然后负载均衡服务器再具体根据某种负载均衡算法机制转发请求到目标的 Web 应用服务器（这些服务器都运行着同样的应用程序），最后将所获得的内容返回到客户端中去。

Nginx 是一个开源、高性能的反向代理服务器，被广泛应用于 Web 应用服务器的负载均衡，提高了 Web 后端服务器集群的资源利用率以及运行效率。Nginx 的负载均衡策略主要有三种：

1) round-robin。这是一种轮询的策略，将 Web 请求以循环、轮转的形式分发到后端的 Web 应用服务器。

2) least-connected。也叫最小连接负载均衡，在这种策略之下，下一个 Web 请求将被分配到拥有最少活动连接数的 Web 应用服务器上。

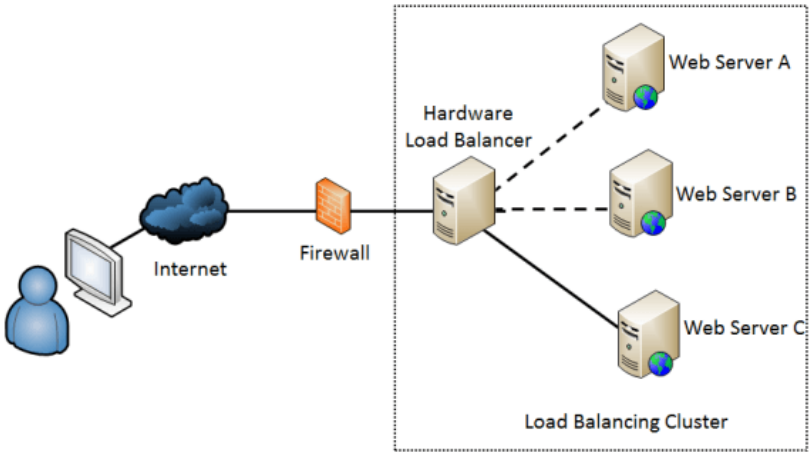

图 3-10 负载均衡示意图

3) ip-hash。这种策略使用一个哈希函数，对客户端的 ip 地址进行哈希映射，从而将该客户端发出的 Web 请求分发到对应的 Web 应用服务器上。

本平台将在后端的 Web 应用服务器集群之上构建一个基于 Nginx 的负载均衡服务器，从而提高平台的性能。

3.4.4 平台架构

综合上述各方面的方案分析，我们给出平台的整体架构图，如图 3-11 所示。

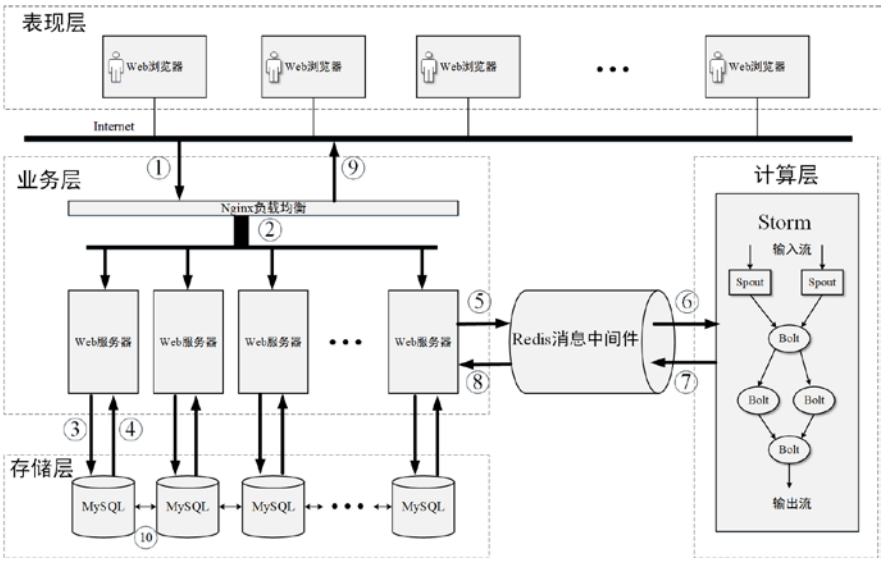

图 3-11 平台整体架构图

图中各个数据流的意义如下：

1) 表现层的用户通过 Web 浏览器与平台进行交互。这个数据流可以是读请求，也就是用户请求对相关数据的查询；也可以是写请求，将病人相关信息录入到平台上或上

传心电数据进行心肌缺血的诊断。

2) 表现层中的所有请求在到达业务层的 Web 应用服务器之前都会经过 Nginx 负载均衡服务器，由它根据特定的负载均衡策略将 Web 请求均衡地分发到 Web 应用服务器中，避免了单点过热的情况，提高了系统的效率。

3) Web 应用服务器与 MySQL 数据库之间的数据交互。同样，该数据流可以是读请求，即 Web 应用服务器根据表现层的具体要求对 MySQL 数据库中的指定数据提出读的请求；也可以是写请求，即 Web 应用服务器将表现层传送过来的数据或计算层传送过来的诊断结果写入 MySQL 数据库中。

4) MySQL 数据库根据具体的读请求将数据返回到 Web 应用服务器中。

5) Web 应用服务器在接收到表现层传来的心电数据后将数据放到 Redis 消息中间件中，以供计算层对数据进行分析处理。

6) 计算层从 Redis 消息中间件中取出心电数据，并将数据作为 Storm 流计算集群的输入数据流，然后对数据进行分析。

7) 计算层将 Storm 集群的分析结果，也就是图中的输出流，放到 Redis 消息中间件中，以供业务层对数据进行进一步的处理。

8) 业务层从 Redis 消息中间件中取出计算层对心电数据的分析结果，并将结果存储到存储层或返回到表现层中。

9) Web 应用服务器根据表现层的数据请求将数据传送到 Nginx 负载均衡服务器，再有它返回到表现层中。

10) MySQL 集群中各个 MySQL 实力之间的数据同步。根据前面的需求分析，本平台的存储层对 MySQL 实力采用的是复制的水平扩展策略，也就是每个 MySQL 都存有一份相同的数据副本。

平台的业务逻辑主要有三种，信息查询、数据写入以及心电数据诊断。每种业务逻辑的具体流程如下：

1) 信息查询。表现层对业务层发出信息查询的 Web 请求，Web 应用服务器收到请求后对 MySQL 数据库发出 SQL 查询，MySQL 再将具体的数据返回到 Web 应用服务器中，最后业务层将数据传送到表现层，而表现层则对数据进行可视化。

2) 数据写入。表现层将需要写入的数据，包括病人的基本信息和病历信息，传送到业务层，Web 应用服务器收到数据后将其发送到存储层中进行持久性存储。

3) 心电数据诊断。表现层上传心电数据到业务层中，Web 应用服务器收到数据后，

首先将原始的心电数据存储到 MySQL 数据库中，然后将心电数据放入 Redis 消息中间件中。计算层一直监听 Redis 消息中间件，当有心电数据达到时，计算层就取出数据作为 Storm 流计算集群的输入数据流，然后对心电数据进行心肌缺血诊断分析，最后将诊断结果输出到输出流中。紧接着，计算层将 Storm 集群的输出流放入 Redis 消息中间件中，当 Web 应用服务器监听到 Redis 消息中间件中存在诊断结果时就从中取出，然后将诊断结果存入 MySQL 数据库并将结果返回到表现层。

3.5 本章小结

本章对平台的设计进行了详细的需求分析和讨论。首先，本章给出了平台三大功能，包括分析计算功能、数据存储功能以及信息管理功能。接着，针对每一个功能进行了具体的需求分析。平台的分析计算功能需要实现对确定学习算法的程序化，以及拥有对海量数据实时计算的能力；数据存储功能必须拥有对海量数据的存储能力以及具备一定的扩展性和容错性；信息管理功能则有并发访问、负载均衡、界面友好、操作简单等需求。完成对平台的需求分析之后，本章着手于方案的选取。在分析计算功能上，本文选择了基于确定学习心肌缺血算法的 C++版实现，并在 Storm 流计算框架上进行计算分析。在数据存储功能上，本文选择了 MySQL 关系型数据库，并通过复制的水平扩展来构建 MySQL 集群。在信息管理功能上，本文在基于 B/S 的架构上构建了 Web 应用程序，其中选择 Boostrap 作为前端的框架，Dropwizard 作为后端的框架。接着，本文对平台的整体架构进行了设计，一共将平台划分为四层，表现层、业务层、计算层以及存储层。最后，对平台的三个主要业务逻辑（信息查询、数据写入和心电数据诊断）的流程进行了简要的描述。

第四章 流式计算框架 Storm 和计算层实现

4.1 引言

根据前一章对计算层的需求分析，本平台最终选取了流式计算框架 Storm 作为计算层的实现框架，并将 C++版的心肌缺血诊断程序运行于 Storm 之上对心电数据进行分析诊断。本章首先对流式计算框架 Storm 进行简单的介绍，然后讨论如何将心肌缺血诊断程序移植到 Storm 框架上，接着介绍 Storm 集群的部署与配置，最后对在 Storm 框架上运行心肌缺血诊断程序的性能进行分析。

4.2 流式计算框架 Storm

4.2.1 Storm 整体架构

Storm 是一个采用主从模式（Master-Slave）架构的流式计算框架，在 Storm 的架构中存在两种类型的节点，主控节点（Master Node）以及工作节点（Worker Node）。主控节点上运行着一个被称为 Nimbus 的后台程序，Nimbus 的主要职责是将流式计算代码以及计算任务分配到各个工作节点上，并对它们进行监控。在一个 Storm 集群中，只有一个主控节点，但存在多个工作节点。工作节点上运行着一个被称为 Supervisor 的后台程序，负责监听 Nimbus 所分配的计算任务，并根据任务的具体要求启动或停止工作进程。Storm 通过 ZooKeeper 集群对 Nimbus 和 Supervisor 之间的工作进行协调与交互，并在 ZooKeeper 集群上存储两者的状态信息，以便在 Nimbus 和 Supervisor 成为无状态服务的时候方便地对故障进行恢复，使得系统具有容错的能力。Storm 的整体架构如图 4-1 所示。

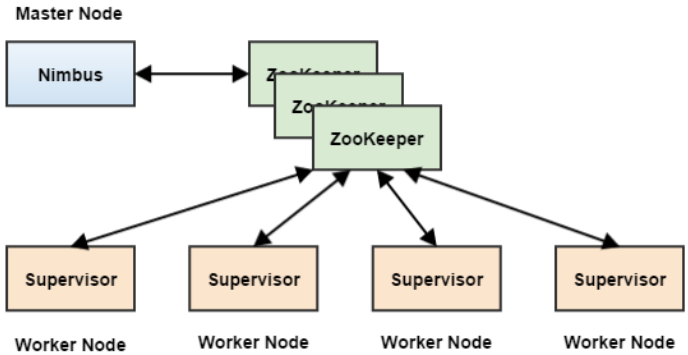

图 4-1 Storm 架构图

4.2.2 分布式协调系统 ZooKeeper

ZooKeeper 是一个高吞吐、可扩展的分布式协调系统，有 Yahoo 公司开发并开源，

目前是 Apache 下的一个子项目。ZooKeeper 主要运用在大规模分布式系统中解决各种类型的协调问题，例如当集群中扩展一台机器后如何进行自动参数配置、当主控节点发生故障时如何从备节点中选出新的主控节点使得系统不至于瘫痪、如何在多个节点之间实现任务同步以及负载均衡等等。目前，ZooKeeper 已经被广泛应用在众多的项目，如 Apache HBase、Apache Kafka 以及 Apache Storm 等。

如图 4-2 所示，ZooKeeper 主要有若干台服务器组成，ZooKeeper 的客户端是应用程序与 ZooKeeper 服务器交互的媒介，ZooKeeper 可以同时响应上万个客户端的请求。每一台 ZooKeeper 服务器都存有类似文件系统的数据结构而且每台服务器都会保存同样的一份数据。ZooKeeper 通过 ZAB 原子广播协议选举一台是主控服务器，而其他的则作为从属服务器。ZooKeeper 采用多数投票仲裁（Majority Quorums）方式来在所有服务器中选取主控服务器，从而保证了系统的一致性。如果一个 ZooKeeper 集群有 2N+1 台服务器，则系统最多可以容忍 N 台故障服务器，因为只有多数投票的服务器存活，ZooKeeper 才能正常地工作运行。

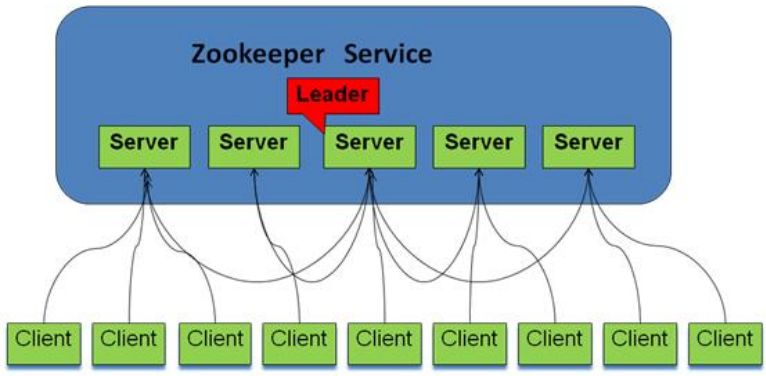

图 4-2 ZooKeeper 架构图

ZooKeeper 的数据模型是一种类似于文件系统的树形数据结构，如图 4-3 所示。每个节点被称为 ZNode，并用“/”来对 ZNode 的名字进行区分，每个 ZNode 由唯一的路径所决定，而且可以存储少量以 Byte 为数据形式的数据。ZNode 包括两种形式，持久化（persistent）节点以及临时（ephemeral）节点。前者只有在客户端调用删除操作时才会消失，而不仅仅只存活在一次会话中。后者着在会话结束后或服务器发生故障时被 ZooKeeper 自动删除。因此，持久化节点主要用于存储类似任务分配信息、运行状态等重要的状态信息，而临时节点则主要用在提示 ZooKeeper 创建它的应用程序还处在活跃的状态。

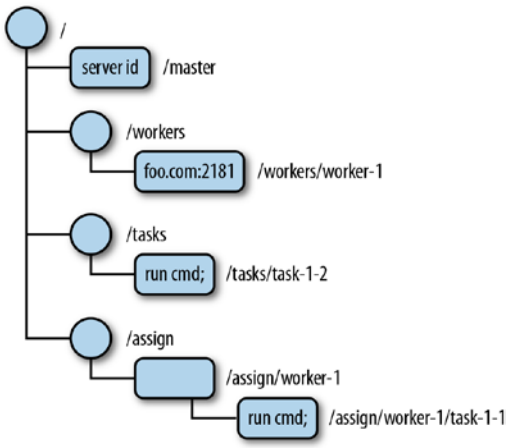

图 4-3 ZooKeeper 数据模型

4.2.3 Storm 的计算模型

在 Storm 框架中，Storm 的计算模型通常都是由多个计算节点构成的有向无环图（DAG）。如图 4-4 所示，一个 Storm 拓扑结构（Topology）主要由数据流（Stream）、流源组件（Spout）以及流处理组件（Bolt）组成。

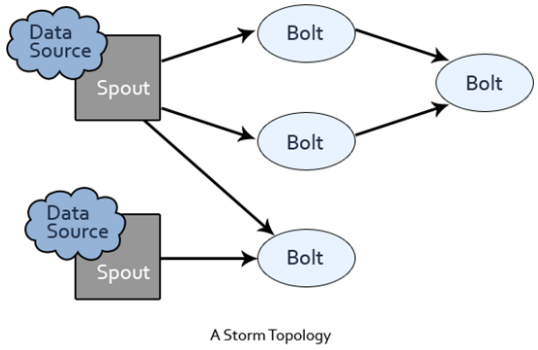

图 4-4 Storm 拓扑结构

其中各个组成部分的功能如下[59]：

1) Stream。Storm 中的核心数据结构被称为元组（Tuple），由一系列的键值对（key-value pairs）所组成。而 Stream 是一个抽象，表示由若干个 Tuple 所组成的无边界的序列。

2) Spout。在一个 Storm 拓扑结构中，Spout 作为数据的输入节点，主要的职责是从数据源中获取数据，然后将数据转换成 Storm 的数据结构 Tuple 并将 Tuple 作为 Stream 提交到下一个节点。

3) Bolt。Bolt 就相当于计算机中的处理器，从 Spout 或 Bolt 输出的 Stream 中获取数据，然后对数据进行处理，最后将结果数据转换成新的 Tuple 并作为 Stream 提交到其他 Bolt 中。

4) Topology。Topology 就是由一系列的 Spout 和 Bolt 通过 Stream 连接在一起形成的 DAG。将代表着计算任务的 Topology 成功提交到 Storm 集群上后，除非显示地将其停止，否则它会一直运行着。

提交到 Storm 集群上的计算任务是并发地在若干个工作节点上执行的，而并发的程度可以通过 Storm 配置文件进行配置。在 Storm 集群里面，工作进程（worker）表示服务器集群的工作节点上的一个 Java 虚拟机（JVM）进程。而执行器（executor）则是包含在 worker 里面的执行具体计算任务的线程。任务（task）是在 Spout 或 Bolt 中定义的处理函数，task 的数量代表着一个 Topology 中每个 Spout 或 Bolt 的并发度。在 Storm 的默认配置中，一个服务器节点中，每个 Topology 有一个 worker 进程，而在每个 worker 进程里面有一个 executor 线程，每个 executor 线程执行一个 task。当然，可以通过配置使得每个节点运行多个 worker 进程，每个 worker 进程里又可以配置多个 executor 线程，而每个 executor 线程又可以并发地执行多个 task，如图 4-5 所示。因此，Storm 流式计算框架在并发性上具有极高的自由度。

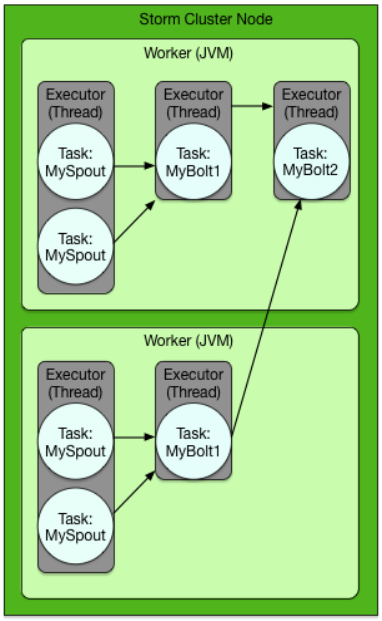

图 4-5 Topology 执行的并发程度

在确定输入输出数据流和各个计算节点的处理逻辑之后，就要设计一个拓扑结构使得将所有的计算节点连接起来后可以表述整个计算任务。在 Storm 中，常见的基本拓扑结构主要有流水线结构、乱序分组结构、定向分组结构以及广播结构。

流水线（pipeline）拓扑结构是最为常见的一种结构，如图 4-6 所示，每个计算节点只有一个输入流和输出流。

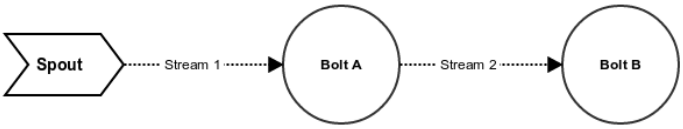

图 4-6 流水线拓扑结构

乱序分组（shuffle grouping）拓扑结构也是较为常用的一种拓扑结构。如果上游的计算节点向下游的计算节点分发数据时采取的是乱序分发策略，则上游计算节点的输出数据将会随机地分发到下游的计算节点中，如图 4-7 所示。这种随机分发的策略是对大数据进行负载均衡的一种较好的机制。

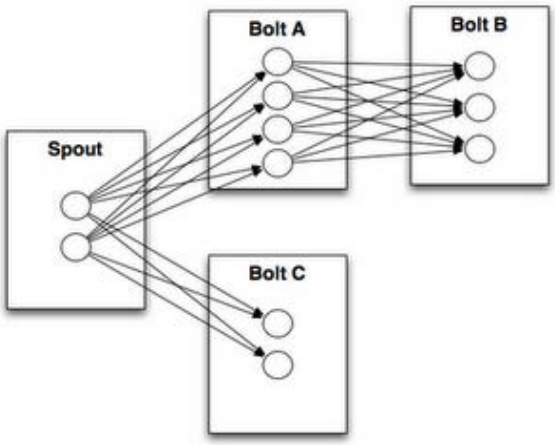

图 4-7 乱序分发拓扑结构

定向分组（field grouping）虽然在体系结构上与乱序分组较为类似，但是它们之间上下游计算节点的分发数据模式并不相同。不同于乱序分组的随机分发，定向分组往往对数据的某一个属性进行哈希映射，从而确保同一属性的数据能够被固定地分发到下游的相对应的计算节点中去。

广播拓扑结构在体系结构上也与乱序分组类似，但它其在分发数据时，上游的计算节点会将同一个数据向下游的所有的计算节点都分发一次。

在众多的流式计算框架中，Storm 提供了最灵活的有向无环图拓扑结构的定义，除了上述几种常见的拓扑连接方式外，Storm 还提供了几种并不常见的拓扑结构。另外，可以通过将各种基础的拓扑结构组合起来形成更为复杂的 DAG 拓扑结构图来解决复杂的计算任务。

4.2.4 Storm 的送达保证机制

Storm 的送达保证机制实现了“至少送达一次”的语义，其运行机制如下：

1) Spout 对每一个输入数据都赋予一个消息 ID 作为该输入数据的唯一标识，这个

ID 是一个 64 位的二进制数。随着输入数据不断地在下游的节点之间被传递，该 ID 也会一直被传递下去。不管数据传递到下游哪个计算节点，从该数据衍生出来的新数据都会记住其都是原始输入数据的后代。

2) 下游的 Bolt 接收到 Spout 传来的输入数据及其消息 ID 后，会对数据进行转换，生成了零个或多个新的数据，并为这些新的数据都赋予一个 64 位的消息 ID。另外，每个新产生的数据都会绑定原始输入数据的消息 ID，以此表明新的数据是由原始输入数据衍生出来的。

3) 假设下游的 Bolt（称之为 N 节点）成功地接收到了原始输入数据或由它衍生出来的数据，在完成对数据的逻辑操作之后，N 节点会在 ACK( )函数中将该节点输入数据的消息 ID、该节点产生的所有新数据的消息 ID 和原始输入数据的消息 ID 进行异或（XOR）操作，用得到的结果将原始输入数据的消息 ID 替换掉。

4) 存在这样的一种情况，一个新的数据是由多个不同的输入数据衍生出来的，此时，多个输入数据的消息 ID 会绑定到新的数据中，以此来表明该数据是由多个不同的数据共同产生的，但是该数据的消息 ID 只有一个。

5) 当下游某个 Bolt 在更新原始输入数据的消息 ID 后，如果消息 ID 的值变为 0，则表明 Storm 已经完成对原始数据的操作，因此不会再将该节点生成的新数据传递到下游节点。此时，该 Bolt 会向传递出该原始输入数据的 Spout 发送 commit 消息，以此表明原始数据已经被成功地处理。

6) 所有的数据，包括原始输入数据及其衍生出来的新数据，和它们所对应的消息 ID 都会被 Storm 存储在一张系统表 T 上。Storm 会对系统表 T 进行定期的扫描，如果发现有一定时间内还没有处理完成的数据（在 T 中表现为数据对应的消息 ID 不为零），则会对数据源 Spout 发出重发数据的消息，以此达到对处理失败的数据进行反复的尝试的目的。

4.3 心肌缺血诊断程序移植

4.3.1 C++版心肌缺血诊断程序

在第二章中，本文对确定学习理论以及其在心肌缺血诊断上的应用，下面将会对基于确定学习的心肌缺血诊断算法的主要流程进行介绍。如图 4-8 所示，算法流程的第一步是要读取原始的心电数据。由于原始的心电数据是标准的十二导联心电图数据，因此算法的第二步将原始的心电数据由十二导联体系转换为 XYZ 三向导联体系，得到心电

向量图（VCG）数据。算法的第三、四步分别对转换后的数据进行中值滤波和小波滤波。接着第五步是选取 VDG 三向导联中的 X 导联，并截取其中的 ST-T 段，对其拼接后形成 ST-T 环。算法的最后一步是运用确定学习理论对 ST-T 环进行学习训练，并将训练的结果绘制成 CDG 环。

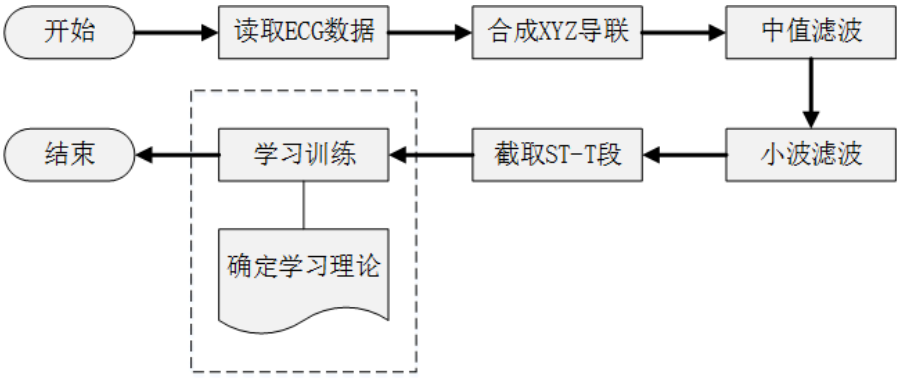

图 4-8 心肌缺血诊断算法流程图

其中，学习训练过程是整个算法的核心，利用确定学习理论对 ST-T 环进行学习训练过程的流程图如图 4-9 所示。首先，对 ST-T 环数据进行归一化处理，从而使其处在一个原点附近的正方体里面。接着，在正方体里面均匀地布置神经元，然后对各个神经元的径向基函数进行计算。最后，训练整个神经网络指导网络权值收敛为止，最终得出的神经网络权值就是 CDG 环的数据。

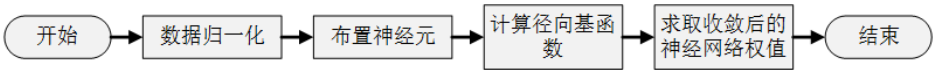

图 4-9 学习训练流程图

在对整个心肌缺血诊断算法进行 C++实现时，需要依赖于如下几个库：Armadillo、OpenBLAS、fftw3 和 wavelet。其中，Armadillo 是一个高性能的 C++线性代数库，提供了类似 Matlab 的高级应用程序接口。它支持整数、浮点数以及复数的计算，并为向量和矩阵实现了高效的类，别广泛应用在机器学习、模式识别、信号处理等领域。OpenBLAS 则是一个多线程的 C++线性代数库，Armadillo 能够在借助 OpenBLAS 的情况下将运算速度提升十倍。因为在心肌缺血诊断算法的流程里需要对数据进行小波变换，因此程序需要依赖于 C++小波变换库 wavelet，而在编译安装 wavelet 之前必须编译安装好快速傅里叶变换库 fftw3。

已有的 C++心肌缺血诊断程序对算法的流程进行了功能模块化，将整个流程分为了以下几个模块：导联体系转换模块、中值滤波模块、小波滤波模块、ST-T 段截取模块、拼接 ST-T 环模块以及学习训练模块。程序将每个模块都封装成一个 C++函数，上述就

一个模块所对应的函数分别是 pretreat 函数、midfilter 函数、waveletfilter 函数、cutST 函数、merge_stt 函数以及 learn 函数。最后将所有的函数封装到一个 main 函数里，构成完整的心肌缺血诊断程序。

原有的 Matlab 版本的心肌缺血诊断程序在对一份心电数据进行学习训练时，耗时大概 60 秒，而 C++版本的心肌缺血诊断程序完成学习训练只需 20 秒。因此后者的计算速度相比前者而言提升了接近 3 倍，提高了算法的运行效率。

4.3.2 Multi-Language 协议

和 Hadoop 一样，Storm 的原生开发语言是 Java，因此，不能直接将 C++版的心肌缺血诊断程序放到 Storm 框架上运行。要想在 Storm 框架上运行非 JVM 语言的程序，必须依靠 Storm 内置的 Multi-Language 协议，其核心思想就是在操作系统的 Shell 中执行非 JVM 语言的可执行程序或脚本。Storm 有专门的类用于支持 Multi-Language 协议，主要是 ShellBolt 类和 ShellSpout 类。在 Storm 中，每个 Spout 和 Bolt 节点都有一个对应的类，这些类都直接或间接继承了 Storm 内置的 ISpout 接口和 IBolt 接口。ShellSpout 类和 ShellBolt 类都分别继承了 ISpout 接口和 IBolt 接口，但与一般的 Spout 类和 Bolt 类不同，数据的处理逻辑并不是直接在这两者上实现。它们只是对非 JVM 语言的可执行文件或脚本的一个包装，真正的处理逻辑是在可执行文件或脚本中实现的。

Multi-Language 协议通过操作系统中的标准输入输出来实现计算节点之间的数据交互。如图 4-10 所示，ShellSpout 或 ShellBolt 包含着实现数据处理逻辑的非 JVM 语言的可执行文件或脚本，可执行文件或脚本对数据处理完成之后通过标准输出（STDOUT）将结果以特定的格式输出到 Shell 中，下游节点监听到有数据后通过标准输入（STDIN）将数据读取，然后对数据进行相应的逻辑处理，最后结果同样通过 标准输出（STDOUT）输出，数据就是这样一路传递下去，直至完成整个计算任务。

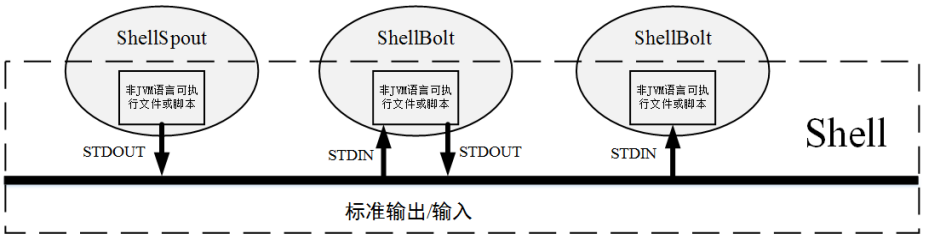

图 4-10 Multi-Language 协议机制

Multi-Language 协议要求通过标准输入/输出进行交互的数据格式必须是 JSON 的格式，这些 JSON 格式的消息可以看成是 Storm 原生的 Stream。JSON 数据格式以一种键-值对的形式存储数据，而 Multi-Language 协议下的 JSON 消息通常包含以下几个键，

command，id，stream 以及 tuple 等。command 键表示这个数据的用途，这个键对应的值通常是 emit（表示一般的数据流）、ack（表示完成数据逻辑处理后的确认消息）、fail（表示处理数据失败的消息）、log（表示系统日志消息）。计算节点的输出数据都存在 tuple 键对应的值上，id 是 tuple 的唯一标识，而 stream 表示该 tuple 提交到哪个 Stream 上。下面给出了一个通过标准输出的消息代码样例：

```
{
    "command": "emit",
    "id": "1231231",
    "stream": "1",
    "task": 9,
    "tuple": ["field1", 2, 3]
}
```

4.3.3 Storm 框架下的心肌缺血诊断程序

根据上述的 C++心肌缺血诊断程序的流程，下面将会首先对它的几个功能模块进行重新的划分，然后再将重新划分后的每个功能实现为 Storm 中的一个 Bolt。

已有的 C++心肌缺血诊断程序将诊断流程分为六个功能模块，包括导联转换模块（pretreat 函数）、中值滤波模块（midfilter 函数）、小波滤波模块（waveletfilter 函数）、截取 ST-T 段模块（cutST 函数）、拼接 ST-T 环模块（merge_stt 函数）以及学习训练模块（learn 函数）。本文将导联转换模块、中值滤波模块和小波滤波模块联合起来组成预处理模块，并将功能实现于 Pretreat Bolt 中；将截取 ST-T 段模块和拼接 ST-T 环模块联合起来组成 ST-T 段截取模块，并将功能实现于 CutST Bolt 中；学习训练模块延续已有的功能模块，并将功能实现于 Learn Bolt 中，如图 4-11 所示。

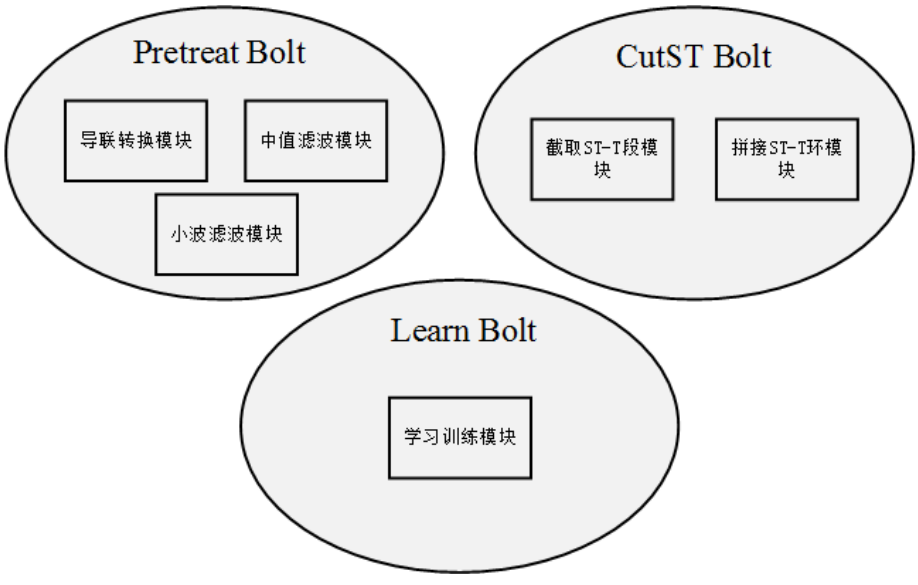

图 4-11 功能模块划分

完成功能模块的重新划分之后，本文下面将讨论各个 Bolt 的具体实现。首先，需要对已有的 C++程序做出修改，使修改后的程序能够运行于 Multi-Language 协议之上。StormCpp 是一个 C++开源组件，实现了 Multi-Language 协议，用于对 C++程序进行包装，使得程序能够运行于 Storm 框架之下。因此，本文将利用 StormCpp 组件对 C++心肌缺血诊断程序进行包装，修改后的程序主要包含三个类，PretreatBolt 类、CutSTBolt 类和 LearnBolt 类。

PretreatBolt 类中实现的是预处理功能，包括导联转换、中值滤波和小波滤波，其代码如下，其中 Process 函数是处理计算逻辑的地方，其输入参数是上游节点传递下来的 Tuple，在完成对 Tuple 中数据的计算逻辑后，将计算结果以 Tuple 的形式提交到下游的计算节点。该类的核心代码如下所示:

```
class PretreatBolt : public Bolt
{
public:
        //初始化函数
        void Initialize(Json::Value conf, Json::Value context) { }
        //处理函数，实现具体逻辑的地方
        void Process(Tuple &tuple) {
                //读取输入数据
                std::string filename = tuple.GetValues()[0].asString();
                std::string data = tuple.GetValues()[1].asString();
                //对心电数据进行滤波
                fmat val(data);
                u32 sfreq = 0, gain = 0,repeat = 0;
                float TS = 0.0;
                pretreat(val, sfreq, repeat, TS, gain);
                //从十二导联中提取出 XYZ 导联
                fmat xyz = val.cols(12, 14);
                //构造输出的 tuple
                Json::Value result(filename, sfreq, xyz);
                Tuple t(result);
                //将 tuple 提交到下游节点
                Emit(t);
    }
};
```

CutSTBolt 类中实现的是 ST-T 段的截取功能以及 ST-T 环的拼接功能，与 PretreatBolt 类类似，该类也是集成 Bolt 类，并且只有两个成员函数，Initialize 负责初始化相关配置，Process 函数负责处理计算逻辑。CutSTBolt 类的代码与 PretreatBolt 类代码类似，不同之处在于 CutSTBolt 类中 Process 函数的功能是对预处理后的数据进行 ST-T 段截取以及 ST-T 环拼接。

LearnBolt 类实现的是对拼接后的 ST-T 环进行学习训练的功能，最后的计算结果是

CDG 环对应的三维矩阵数据。这个过程是对确定学习理论的实现，也是整个心肌缺血诊断过程中最核心的步骤，其代码与 PretreatBolt 类和 CutSTBolt 类类似，不同之处在于 Process 函数的功能是对 CutSTBolt 的输出数据进行基于确定学习的学习训练，并最终得出 CDG 数据。

在完成使用 StormCpp 对原来的 C++程序进行包装后，需要将上述的几个类编译成可执行文件。然后，在可执行文件之上包装一个 Storm 原生的 Java 类，ShellBolt，当有数据流到达计算节点后由 ShellBolt 类调用可执行文件，对数据进行处理。Tuple 是一种键值对类型的数据结构，ShellBolt 类的一个职责是对输出数据定义一个键，使得输出的数据符合键值对的格式。ShellBolt 类构造函数的参数是运行可执行文件的 Shell 命令的字符串形式，而本文通过 Python 脚本间接执行 C++可执行文件，因此 ShellBolt 类中构造函数的参数是执行 Python 脚本的 Shell 命令。与 C++程序中的三个 Bolt 类对应，Java 程序中也定义了三个 ShellBolt 类，分别是 PretreatShellBolt 类、CutSTShellBolt 类和 LearnShellBolt 类。由于三者的代码大致一样，因此下面只给出了 LearnShellBolt 类的核心代码：

```
public class LearnShellBolt extends ShellBolt implements IRichBolt {
        //构造函数，参数为执行 Python 脚本的 Shell 命令
        public LearnShellBolt() {
                super("python", "learn.py");
        }
        ...
    //为输出数据定义键。
        public void declareOutputFields(OutputFieldsDeclarer declarer) {
                declarer.declare(new Fields("testId", "CDG"));
        }
        ...
}
```

4.3.4 拓扑结构设计

上一小节介绍了在 Storm 中三个计算节点的实现，这三个计算节点分别是 Pretreat Bolt、CutST Bolt 和 Learn Bolt。接下来，本文将会对剩余两个节点的实现进行介绍，分别是读取输入数据的 Read Redis Spout 节点和输出结果数据的 Write Redis Bolt 节点。

由第三章的平台架构设计可知，计算层与业务层之间的数据交互是通过 Redis 消息中间件进行的。业务层把需要计算的心电数据放到 Redis 消息中间件中，计算层再从中取得数据并在 Storm 框架上进行计算，最后将计算结果数据放到 Redis 消息中间件中返回给业务层。因此，本文设计了一个读取输入数据的 Spout 节点，其一直监听 Redis 消

息中间件，当有新的心电数据到达时，就从中取出，对数据进行转换后以 Tuple 的形式提交到下游节点。ReadRedisSpout 类是对该 Spout 节点的具体实现。

Write Redis Bolt 节点的职责是接收上游节点提交过来的 CDG 数据，然后对数据进行加工，并放到 Redis 消息中间件中，以供业务层对计算结果数据进行进一步的处理。WriteRedisBolt 类是该 Bolt 节点的具体实现。

到目前为止，本文为计算层设计了五类计算节点，按照计算过程的先后顺序，分别是 Read Redis Spout 节点、Pretreat Bolt 节点、CutST Bolt 节点、Learn Bolt 节点和 Write Redis Bolt 节点。由于各个节点之间的计算逻辑相对独立，只需保证各类节点的先后顺序即可。根据 4.2 节的分析，乱序分组拓扑结构的实现较为简单，而且能够对计算节点进行负载均衡，因此本文选取了乱序分组作为计算层的拓扑结构，如图 4-12 所示。每个节点在完成对数据的处理之后，都将结果随机地分发到下游的节点。整个拓扑结构的实现被放在了 CardioTopology 类中，核心代码如下：

```
public class CardioTopology {
        public static void main(String[] args) throws Exception {
                ...
                //构建拓扑结构
                TopologyBuilder builder = new TopologyBuilder();
                //设置节点的先后顺序以及分组策略，这里为乱序分组
                builder.setSpout("spout", new ReadRedisSpout());
                builder.setBolt("pretreat", new PretreatBolt())
                        .shuffleGrouping("spout");
                builder.setBolt("cutST", new CutSTBolt())
                        .shuffleGrouping("pretreat");
                builder.setBolt("learn", new LearnBolt())
                        .shuffleGrouping("cutST");
                builder.setBolt("save", new WriteRedisBolt())
                        .shuffleGrouping("learn");
                ...
                //将拓扑提交到 Storm 集群上
                StormSubmitter.submitTopology("cardio", conf,
                                builder.createTopology());
        }
}
```

4.4 Storm 集群搭建与性能分析

4.4.1 Storm 集群搭建

Storm 集群是平台的核心，使得平台能够为医务人员以及相关领域的研究人员提供并发高效的计算服务。下面本文将根据实验室已有的硬件设施，完成对 Storm 集群的搭建，并在集群上部署前文所述的心肌缺血诊断程序。

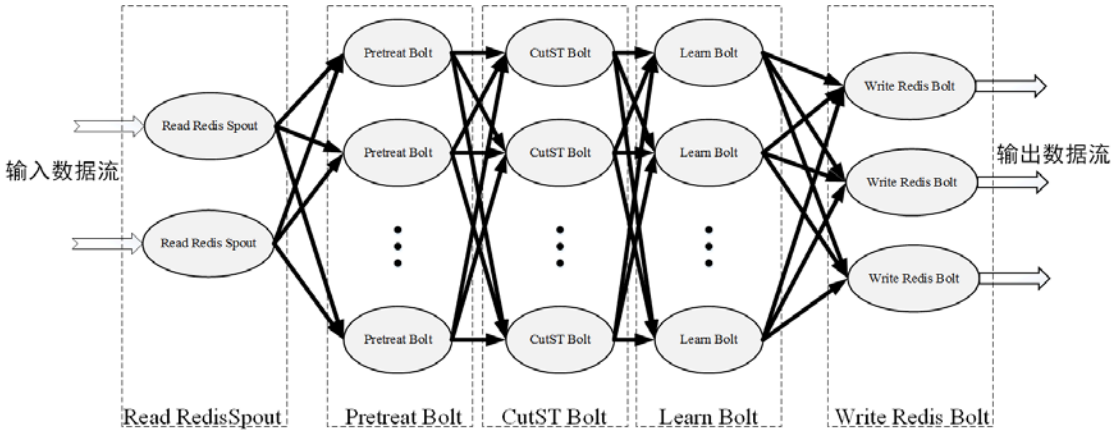

图 4-12 心肌缺血诊断拓扑结构

本人所在实验室目前已经购置五台华为公司的服务器，其中一台配置较高，另外四台配置相对较低。配置较高的服务器型号为 Tecal RH2288H V2，该服务器配备双路四核 CPU Intel Xeon CPU E5-2609 v2@2.50GHz、64GB DDR3ECC 四通道内存、3TB SATA 固态硬盘。配置较低的四台服务器型号为 RH2288 V3，该服务器配备单路六核 CPU Intel®Xeon®CPU E5-2620 v3@2.40GHz、32GB DDR4ECC 内存、2TB SATA 固态硬盘。五台服务器都装有 CentOS 操作系统，并预先进行了 IP 地址配置、JDK 安装、SHH 配置等基本的准备操作。本文选用其中的四台来搭建 Storm 集群，其中配置较高的一台服务器作为 Storm 集群的主控节点 Nimbus，命名为 master；剩余三台作为 Storm 集群的工作节点，并在上面搭建 ZooKeeper 集群，分别命名为 slave1、slave2 和 slave3。如图 4-13 为 Storm 集群的规划图。

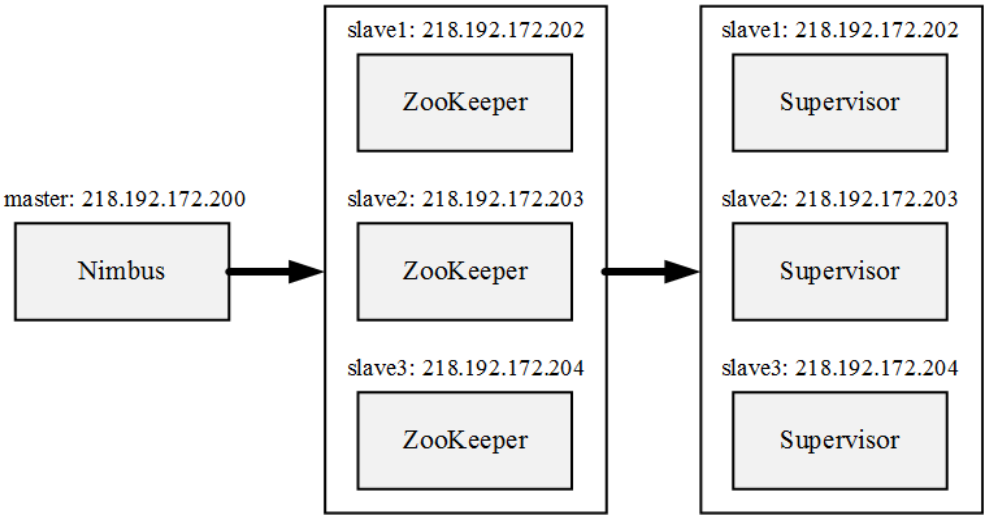

图 4-13 Storm 集群规划图

Storm 集群的安装过程如下：

(1) ZooKeeper 集群安装。从官网下载最新的 ZooKeeper 安装包到 slave1、slave2 和 slave3 中，解压后添加 ZOOKEEPER_HOME 环境变量，并将 ZooKeeper 目录下的 bin 目录添加到系统路径 Path 下。修改配置文件 conf/zoo.cfg 如下：

```
#数据文件目录
dataDir=/var/data/zookeeper
#ZooKeeper 集群对应的 ip 地址和端口（server.id=host:port:port）
server.1=slave1:2888:3888
server.2=slave2:2888:3888
server.3=slave3:2888:3888
```

(2) ZeroMQ 安装。ZeroMQ 是一个并发的 C++网络链接库，常用于在多种协议中进行消息的传输，如进程间通信、线程间通信、TCP 协议等。ZeroMQ 有足够快的传输速度来胜任集群中的应用，它的异步 I/O 机制使其能够用于构建多核的应用程序。Storm 集群中的节点就是通过 ZeroMQ 进行信息或数据传输的。从官网下载最新的 ZeroMQ 安装包到每台服务器上，解压，使用 make 进行编译安装即可。

(3) JZMQ 安装。ZeroMQ 是 C++的链接库，而 Storm 是在 JVM 中运行的，因此 Storm 无法直接使用 ZeroMQ。JZMQ 是使用 Java 原生接口（Java Native Interface，JNI）对 ZeroMQ 进行封装的 Java 库，Storm 需要通过 JZMQ 来使用 ZeroMQ。安装过程与 ZeroMQ 的安装过程类似，从 github 上下载最新的 JZMQ 安装包到各台服务器上，解压后使用 make 编译安装。

(4) Storm 安装。从而官网下载最新的 Storm 安装包到各台服务器上，解压后配置 STORM_HOME 环境变量，并将 Storm 目录下的 bin 目录添加到系统路径 PATH 中。然后修改配置文件 conf/storm.yaml，下面列出了几个重要的配置项：

a) storm.zookeeper.servers：用于指定 ZooKeeper 服务器的地址，其在 storm.yaml 中的配置内容如下：

```
storm.zookeeper.servers:
 -  "218.192.172.202"
 -  "218.192.172.203"
 -  "218.192.172.204"
```

b) storm.local.dir 用于指定存储 Nimbus 和 Supervisor 守护进程的一些状态信息的目录，其配置内容如下：

```
storm.local.dir: "/var/data/storm"
```

c) nimbus.host 用于指定主控节点的 ip 地址，因为 Supervisor 需要知道哪台服务器是 Nimbus，从而在 Nimbus 上下载 Topology 的 jar 包以及配置文件，其配置内容如下：

```
nimbus.host: "218.192.172.200"
```

d) supervisor.slots.ports 用于指定一个 Supervisor 上所使用的端口，每一个 worker 都使用一个单独的端口来接发数据，因此通过该项可以配置一台服务器上运行的 worker 数目。默认情况下，storm 中的每个节点服务器会有四个 worker 运行在 6700~6703 的四个端口上。该项的配置内容如下：

```
supervisor.slots.ports:
   - 6700
   - 6701
   - 6702
   - 6703
```

完成 Storm 集群的安装后，需要执行以下命令来启动集群。

首先在主控节点上执行命令：

```
nohup storm nimbus &
```

在工作节点上执行命令：

```
nohup storm supervisor &
```

需要注意的是，每台 Supervisor 节点上的四个 worker 并不会在节点启动时就立即启动，它们会在根据 Nimbus 分配过来的任务按需启动。也就是 worker 的启动是由提交到 Nimbus 的 Topology 决定的，如果 Topology 只指定了一个 worker 执行任务，那么 Supervisor 节点上就只会启动一个 worker。

4.4.2 性能分析

本文所述平台的计算层是基于 Storm 流式计算框架对数据进行分布式计算的，Storm 以集群的形式部署，可以充分利用多台服务器的计算能力从而提升了计算的速度。本文将针对 Storm 框架上的诊断程序与原有的 Matlab 版的诊断程序做性能对比。性能测试所用的数据是从迈瑞心电图机上采集的标准十二导联心电数据，并将测试数据分成 6 组，样本数分别是 1、10、20、50、100、200。Storm 集群的硬件为前文所述的五台服务器；由于 Matlab 版的诊断程序是基于 Windows 系统开发的，因此其硬件设施为一台配备单路四核 Intel Core i7-4790 CPU @3.6GHz 处理器、16GB DDR3 内存、装有 Windows 10 操作系统的戴尔笔记本。

在进行 Storm 集群的性能测试时，对需要提交到 Nimbus 上的 Topology 进行并发程度的设置，其中 Read Redis Spout 节点的线程数设为 10，即一个 worker 里会有两个 Spout 线程进行数据分发；Pretreat Bolt、CutST Bolt 和 Learn Bolt 节点的线程数设为 20，任务

数也为 20，即一个 worker 中有 20 个线程，每个线程里有一个 Bolt 实例；Write Redis Bolt 节点的线程数设为 10，任务数也为 10。

对 Matlab 版的心肌缺血诊断程序进行性能测试的代码如下，其中 diagnosis 函数是对诊断算法的实现函数。

```
array = [1,10,20,50,100,200];
duration = [];
for i = 1:length(array)
    times = array(i);
    tic;
    for j = 1:times
        diagnosis('./data/','966123_20160904966123.txt');
    end
    duration = [duration,toc];
end
duration
```

两者性能测试的对比结果如图 4-14 所示。从图中可以看书，Matlab 版诊断程序的计算时间与输入心电数据的数量基本成线性增长关系。而在 Storm 集群上对心电数据进行诊断的速度明显加快，并且在数据量越大，其对 Matlab 版的诊断程序的加速比越大。另外，在测试用例较小时（20 个以内），由于 Storm 集群有足够的计算资源，因此计算时间大致相等。当测试用例较大时，受限于测试程序对 Topology 并行程度的设置，节点之间的数据流出现排队的情况，因此计算时间有所增加。

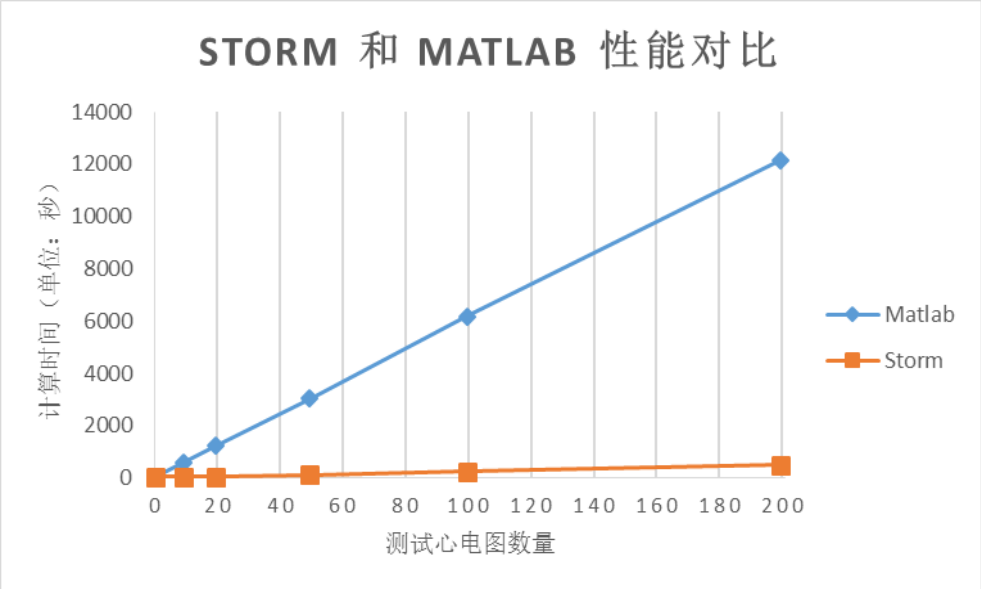

图 4-14 Storm 和 Matlab 性能对比图

根据上述的性能测试对比，基于 Storm 框架的心肌缺血诊断程序相比于原来的 Matlab 版的诊断程序，在对于大量心电数据进行诊断分析时有着明显的优势。

4.5 本章小结

本章详细介绍了平台的基于 Storm 流式计算框架的计算层的实现。首先，本文简要介绍了 Storm 流计算框架的基本知识，包括 Storm 的整体架构、ZooKeeper 分布式协调系统、Storm 的计算模型和 Storm 的送达保证机制。接着，回顾了已有的 C++版心肌缺血诊断程序，并详细描述了将已有的 C++诊断程序移植到 Storm 框架上的过程。基于 Storm 框架的心肌缺血诊断程序一共有 5 类节点，分别是用于读取输入数据的 Read Redis Spout 节点、对数据进行预处理的 Pretreat Bolt 节点、截取 ST-T 段和拼接 ST-T 环的 CutST Bolt 节点、对 ST-T 环采用确定学习算法进行学习训练的 Learn Bolt 节点以及输出诊断结果的 Write Redis Bolt 节点。其后，本文对诊断程序的计算拓扑结构进行了讨论，选择了乱序分发作为最终的拓扑结构。完成 C++心肌缺血诊断程序向 Storm 框架的移植后，本文开始着手搭建 Storm 集群，期间描述了相关软件的安装以及相关配置文件的修改。最后，本文设计了基于 Storm 框架的诊断程序与原来的基于 Matlab 的诊断程序的性能对比实验，实验结果表明，Storm 框架极大地提高了算法对大量心电数据进行诊断分析的运行速度。

第五章 平台的整体实现

5.1 引言

前一章介绍了平台计算层的实现，本章将对平台的整体实现进行详细的介绍。首先，介绍基于 MySQL 数据库存储层的实现，包括数据库表的设计、存储过程的编写以及 MySQL 集群的部署。接着，本章将详细讨论业务层的具体实现，包括业务逻辑的实现、负载均衡配置等。最后，本章将描述平台表现层的设计和开发，并对平台的相关使用流程进行简单的介绍。

5.2 存储层的实现

5.2.1 数据表的设计

存储层主要存储的数据主要包括以下几种：（1）病人基本信息，包括病人姓名、性别、年龄等；（2）病历信息，包括病人的主诉、医生的诊断信息等；（3）病人的原始心电数据；（4）采用基于确定学习的心肌缺血诊断结果。每种数据之间都有着密切的关联，通常，采用实体-联系图（Entity Relationship Diagram，E-R 图）能够准确地描述数据间的关系。E-R 图能够实现对现实世界的概念模型进行描述，常用于数据库逻辑结构的设计。存储层的 E-R 图如图 5-1 所示。

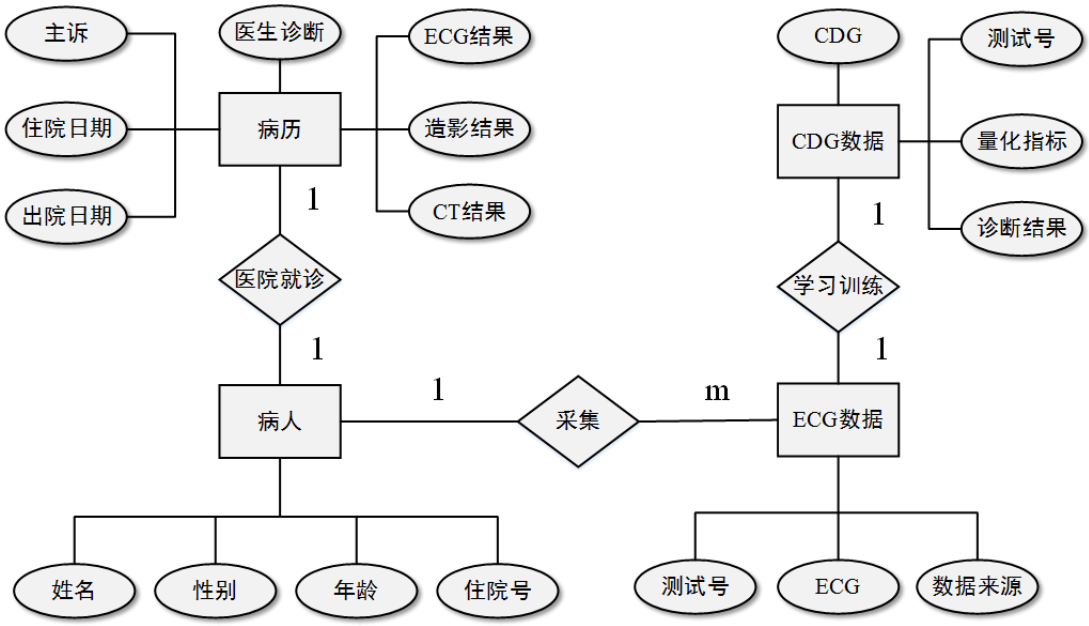

图 5-1 存储层 E-R 图

根据存储层的 E-R 图，本文一共设计五个数据表，分别是 patients 表、cases 表、ecg 表、cdg 表和 times 表。

patients 表存储病人的基本信息，主要的字段以及字段的描述如表 5-1 所示。

表 5-1 patients 表结构

字段名	数据类型	描述	主键
id	int	病人的唯一标识	是
name	varchar	姓名	否
sex	varchar	性别	否
age	tinyint	年龄	否
admissionnumber	varchar	住院号	否

cases 表存储病人的病历信息，主要的字段以及字段的描述如表 5-2 所示。其中 patient_id 是 patients 表中 id 字段的外键；ecg_tag 是 ECG 结果的类型，0 表示 ECG 正常，1 表示异常。ct_tag 和 radiography_tag 分别表示 CT 结果和造影结果的类型，0 表示暂缺，1 表示未见异常，2 表示狭窄。

表 5-2 cases 表结构

字段名	数据类型	描述	主键
id	int	病历的唯一标识	是
patient_id	int	病人 id	否
complaint	text	主诉	否
diagnosis	text	医生诊断	否
ecg	text	ECG 结果	否
ecg_tag	tinyint	ECG 结果类型	否
ct	text	CT 结果	否
ct_tag	tinyint	CT 结果类型	否
radiography	text	造影结果	否
radiography_tag	tinyint	造影结果类型	否
hospitalize_time	date	入院时间	否
discharge_time	date	出院时间	否
disease	tinyint	疾病类型	否
remarks	text	备注	否

ecg 表存储病人的心电数据，主要字段及其描述如表 5-3 所示。其中，testid 为心电数据的测试编号，是该数据的唯一标识；patient_id 字段为 patients 表中 id 字段的外键；ecg_data 为原始的心电数据，以 JSON 格式存储；source 为数据来源，例如迈瑞心电图机。

表 5-3 ecg 表结构

字段名	数据类型	描述	主键
testid	varchar	心电数据的测试编号	是
patient_id	int	病人 id	否
ecg_data	longtext	心电数据	否
source	varchar	数据来源	否

cdg 表存储病人的诊断数据，主要字段及其描述如表 5-4 所示。其中，testid 既是 cdg

表的主键，又是 ecg 表中 testid 的外键；cdg_data 为 CDG 环的数据，以 JSON 格式存储；para_fft 和 para_lya 为量化指标，用于对心电数据的诊断；cdg_results 为根据 CDG 对心电数据进行诊断的结果，有三种类型，阴性、阳性以及可疑阳性。

表 5-4 cdg 表结构

字段名	数据类型	描述	主键
testid	varchar	心电数据的测试编号	是
cdg_data	longtext	CDG 数据	否
cdg_results	varchar	CDG 诊断结果	否
para_fft	double	量化指标	否
para_lya	double	量化指标	否

times 表存储病人采集心电数据的次数，其字段及其描述如表 5-5 所示。其中 patient_id 字段为 patients 表中 id 字段的外键。

表 5-4 cdg 表结构

字段名	数据类型	描述	主键
patient_id	int	病人 id	是
times	tinyint	ECG 测试次数	否

这五个数据表中，patients 表、cases 表、ecg 表和 times 表通过 patients 表中的 id 字段联系起来，而 ecg 表和 cdg 表则通过 testid 字段联系起来，这样就保证了各个数据表中数据存储的正确性。这些表之间的关系如图 5-2 所示。

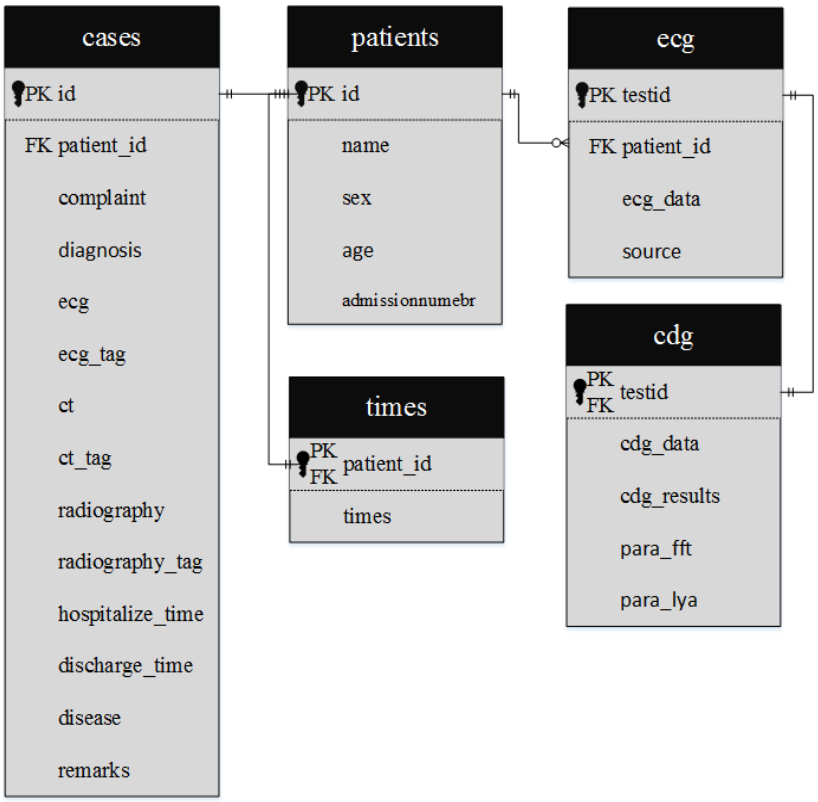

图 5-2 表间关系图

5.2.2 存储过程的设计

存储过程（Stored Procedure）是一组为了完成特定功能的 SQL 语句的集合，它的一大优点就是经过第一次编译后无需经过再次编译即可完成调用，使得其能够快速高效地执行。除了提高数据查询速度外，存储过程还有减少网络流量和提高系统的安全性的作用。因此，存储过程在数据库系统中扮演着一个重要的角色。根据平台的业务需求，本文为存储层设计了两个存储过程，分别是 patientInfoQuery 和 cdgQuery。

存储过程 patientInfoQuery 主要用于根据病人姓名、性别、心电图诊断类型、造影结果类型、CT 结果类型、疾病类型等条件筛选病人。由于这些筛选条件来自不同的数据表，因此不能通过传统的单表查询完成数据的筛选，必须借助多表联结的方式。根据前文所述存储层各个表之间的关系，利用病人的 id 字段和心电图的 testid 字段能够通过内联结将各个表联合起来查询数据。patientInfoQuery 存储过程的核心 SQL 语句如下所示。

```
SELECT DISTINCT patients.name, patients.sex, patients.age,
    patients.admissionnumber, cases.hospitalize_time, ecg.source
        FROM patients INNER JOIN cases ON patients.id = cases.patient_id
            INNER JOIN ecg ON patients.id = ecg.patient_id
            INNER JOIN cdg ON ecg.testid = cdg.testid
            INNER JOIN times ON patients.id = times.patient_id
                WHERE ...
```

存储过程 cdgQuery 是根据病人的 id 来查询相关的 CDG 诊断信息以及基本信息，这也涉及到多表联结，联结的方式与存储过程 patientInfoQuery 类似，其核心的 SQL 语句如下所示。

```
SELECT patients.name, patients.sex, patients.age,
    patients.admissionnumber, cdg.testid, cdg.cdg_results
        FROM patients INNER JOIN ecg ON patients.id = ecg.patient_id
            INNER JOIN cdg ON ecg.testid = cdg.testid
                WHERE  patients.id = id;
```

5.2.3 MySQL 集群的搭建

由第三章的需求分析可知，本文所述平台对 MySQL 数据库选取了主复制的水平扩展方案。我们将在搭建 Storm 集群的四台服务器中选取配置较高的一台作为主数据库 master，其他三台作为被数据库 slave1、slave2 和 slave3。

如图 5-3 所示，MySQL 数据库之间的复制过程实际上是一个异步的过程，这一过程分为三个步骤。首先，Master 数据库将数据的更改记录写入二进制日志（Binary Log）中。然后，Slave 数据库中的 I/O 线程会将 Master 数据库的二进制日志复制到自己的中

继日志（Relay Log）中。最后，Slave 数据库中的 SQL 线程从中继日志中读取事件并在 Slave 数据库中执行，从而实现了主备数据库之间的数据复制。

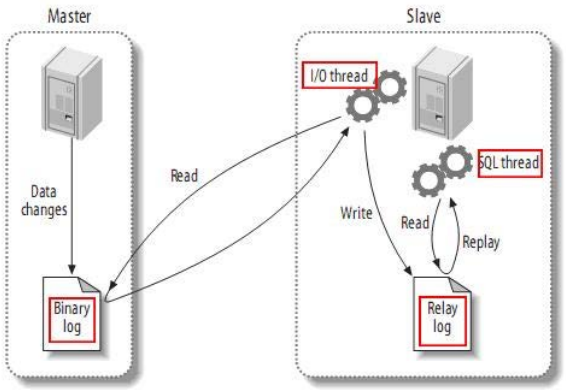

图 5-3 MySQL 复制过程

MySQL 集群的搭建步骤如下：

（1）MySQL 数据库安装。从官网下载最新的 MySQL 安装包到各个服务器上，解压安装，并配置用户名和密码即可。

（2）配置主数据库。修改 master 数据库上的/etc/my.cnf 文件如下所示。

```
#数据库 ID 号， 为 1 时表示为 Master
server-id=1
#启用二进制日志;
log-bin=mysql-bin
#需要同步的二进制数据库名;
binlog-do-db=cardio
#不同步的二进制数据库名;
binlog-ignore-db=mysql
#把更新的记录写到二进制文件中;
log-slave-updates
#跳过错误，继续执行复制;
slave-skip-errors
```

（3）配置备数据库。修改 slave 数据库上的/etc/my,cnf 文件如下所示，由于 slave1、slave2 和 slave3 的修改大致相同，下面只给出了 slave1 的配置内容。

```
server-id=2
log-bin=mysql-bin
#master 数据库 ip 地址
master-host=218.192.172.202
master-user=root   #mysql 用户
master-password = ******   #mysql 用户密码
#如果发现主服务器断线，重新连接的时间差;
master-connect-retry=60
replicate-do-db=cardio
replicate-ignore-db=mysql
log-slave-updates
```

```
slave-skip-errors
```

（4）启动 MySQL 集群。在各台服务器上重新启动 MySQL 后，MySQL 就会根据配置文件的内容实现数据库之间的主从复制。值得注意的是，主从数据库之间所用的登录用户名、用户密码以及需要同步的数据库名必须一致。

5.3 业务层的实现

5.3.1 Dropwizard 框架

Dropwizard 是一个支持 RESTful API 的 Java 微服务框架，它能够为 Web 应用程序提供高效、可靠的实现。平台的业务层中的 Web 应用服务器就是在 Dropwizard 框架的基础上进行业务逻辑的开发的。如图 5-4 所示，Dropwizard 框架主要由 Core、Client、Web 和 Data Access 四大模块组成。在业务层的开发过程中，我们主要用到了 Core 和 Data Access 两个模块。

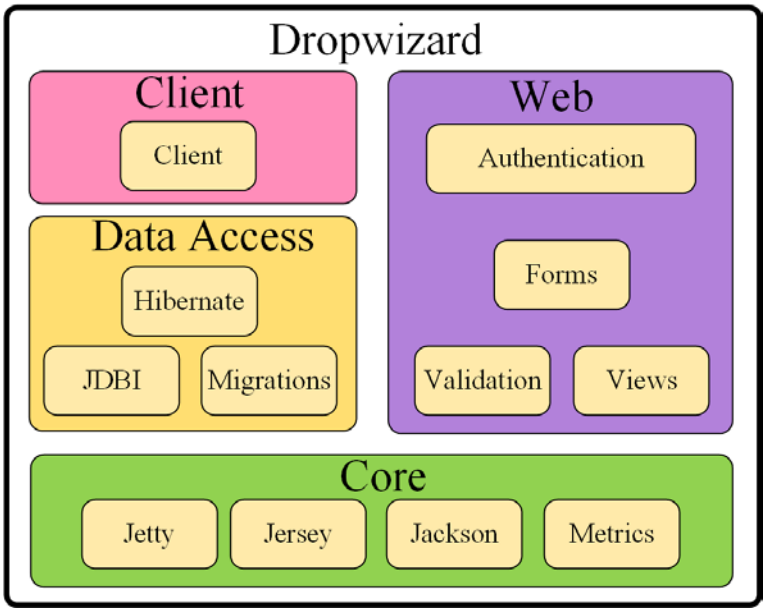

图 5-4 Dropwizard 框架模块组成

Core 模块主要由四个组件组成，分别是 Jetty、Jersey、Jackson 和 Metrics。所有的 Web 应用程序都离不开 HTTP，Dropwizard 框架使用 Jetty HTTP 库嵌入到自身的 HTTP server 上以供开发人员使用。Dropwizard 提供一个 main 函数来启动 HTTP server，而应用程序的业务逻辑的实现都是建立在 HTTP server 之上的。Jersey 库用于支持 RESTful API，它支持流输出、矩阵 URL 参数、条件 GET 请求等多种功能，使得开发者能够将 HTTP 请求映射成一个 Java 对象，从而开发出简洁、可测试的 Web 应用。JSON 是 Web 应用常用的用于交互的数据格式，而 Jackson 是 Java 中最流行的 JSON 库，它有轻量、

速度快等特点。Metrics 库为开发者提供了一个可以监控 Web 应用程序的方法，通过 Metrics，开发者可以时刻知道 Web 应用的运行状况。

在 Data Access 模块中，本文所述平台主要用到了 JDBI 组件。JDBI 是一个方便的 Java SQL 库，用于在 Java 程序中连接关系型数据库，并对其中的数据进行一系列的操作。它提供了面向对象风格的 SQL 操作，使得程序更加容易理解，更具可维护性。

5.3.2 业务逻辑的实现

业务层是可以看成是整个平台系统的枢纽、信息的中转站，不管是数据的上传、存储、查询或者分析计算，数据都要经过业务层的处理，再有业务层分发到其他的各个层次。因此，业务层的业务逻辑的设计，关系着整个平台运行的可靠性与正确性。本文选择 Dropwizard 框架作为业务层的基础框架，并在该框架的基础上完成 Web 应用程序的开发。

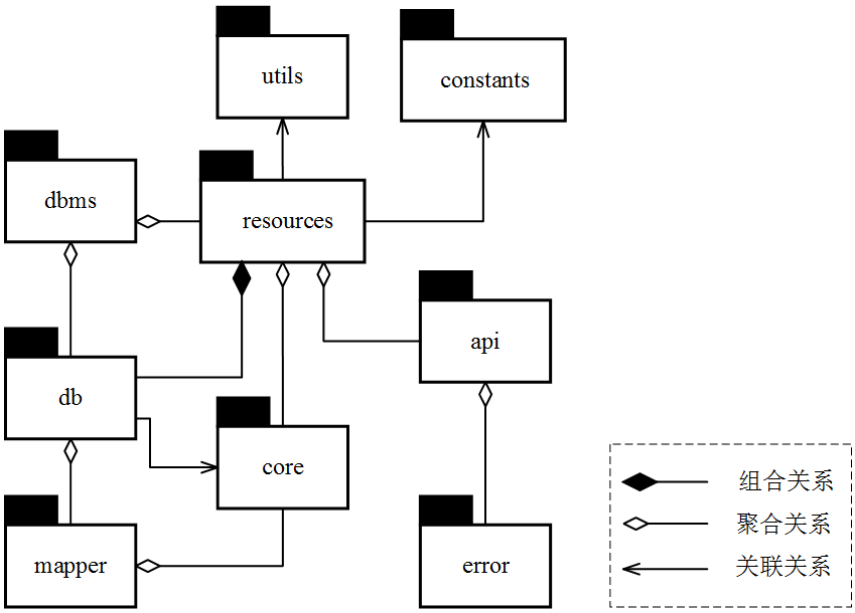

图 5-5 业务层包之间的关系图

业务层 Web 应用程序的各个包（package）及其之间的关系如图 5-5 所示。有图可知，这些包之间共有三种关系：关联关系、聚合关系和组合关系。在这三种关系中，关联关系表示的关系最弱，表示的是语义级别的一种强依赖关系，它不存在依赖关系的偶然性和临时性，一般是长期性的关系。聚合关系的关系强度比关联关系的要强，其实它是关联关系的一种特例，体现的是整体与部分的关系，也就是 has-a 的关系。在聚合关系里，整体和部分之间是可分离的，它们都有独立的生命周期。组合关系在这三种关系里强度最强，它也是关联关系的一种特例，是一种比聚合更强的关系，因此也被称为强聚合。它同样表示的是整体与部分之间的关系，但是整体和部分是不可分离的，整体生

命周期的结束意味着部分生命周期的结束。

各个包的具体描述如下所示：

（1）dbms 包。dbms 包是其他包的父包，其包含两个类，DBMSApplication 类和 DBMSConfiguration 类。其中，DBMSApplication 类是整个 Web 应用程序的核心类，该类提供一个 main 函数用于启动 Dropwizard 的 Jetty HTTP server。此外，Web 应用程序的 RESTful API 都必须在此类中注册才能被其他应用程序调用。DBMSConfiguration 类则用于加载配置文件，使得 Web 应用程序能够按照配置文件的内容正确地运行。Dropwizard 的配置文件为.yml 格式文件，主要是对 Web 应用程序的端口、连接数据库的用户名密码、日志文件目录等内容进行配置，而且改变配置内容并不需要对整个工程进行重新的编译。

（2）core 包。core 包里的类将数据库中的每一个数据表都表示成一个 Java 对象，使得程序能够方便地对数据表进行操作。与存储层中的数据表对应，core 包里一共有五个类，分别是 Patients 类、Cases 类、ECG 类、CDG 类和 Times 类。

（3）mapper 包。mapper 包中的类将对数据库进行 SQL 语句查询返回的结果映射成一个 Java 对象，即 core 包中的各个类的实例。这个映射过程发生在 Mapper 类中的 map 函数中。与 core 包中的类相对应，mapper 包里一共有五个类，分别是 PatientsMapper 类、CasesMapper 类、ECGMapper 类、CDGMapper 类和 TimesMapper 类。

（4）db 包。db 包把对数据表中的 SQL 查询操作映射成 Java 接口中的函数，与 core 包对应，db 包也有五个接口，分别是 PatientsDAO、CasesDAO、ECGDAO、CDGDAO 和 TimesDAO。映射的过程通过 Java 中的注解完成，只读 SQL 查询操作在@SqlQuery 注解中完成映射，而对数据库中数据有所改动的 SQL 查询操作则在@SqlUpdate 注解中完成映射。

（5）utils 包。utils 包里的类都是通用的工具类，例如对文件的读写、对数据进行格式转换等。将这些工具类封装到一个包里，使得工程的中的功能分类更加清晰明了。

（6）constants 包。与 utils 包类似，constants 包将程序常用的一些常量封装起来，这样主要是考虑到程序的可读性。

（7）error 包。在 error 包中，我们将 Java 中基础的异常类封装起来以提供更加具体的异常功能，另外还将一些常用的错误返回码封装到一个枚举类 ErrorCode 中，根据不同的业务错误返回不同的错误码。

（8）api 包。api 包里封装的是 HTTP 请求的响应类，针对不同的 HTTP 请求返回

不同的对象。api 包里一共有四个 Java 类，分别是 Response、InsertUpdateResponse、UploadFileResponse 和 StoreCDGResponse。各类属性、方法以及继承关系如图 5-6 所示。

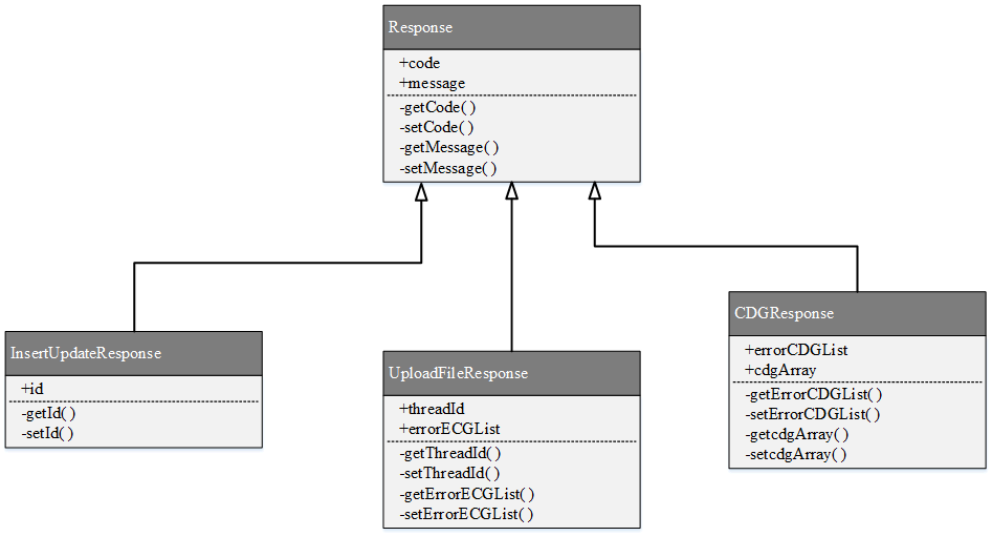

图 5-6 api 类 UML 图

（9）resources 包。resources 包中的类用于实现 Web 应用程序的 RESTful API，通常也被称为资源类。常用的 HTTP 请求方法有四种，分别是 GET、POST、PUT 和 DELETE。GET 请求通常是只读请求，用于请求指定的页面信息并返回实体主体；POST 请求则通常将数据包含在请求中，并向指定的资源提交并进行处理，例如上传文件和提交表单；PUT 请求用于向服务器传送数据并替代指定的文档内容；DELETE 请求用于请求服务器删除指定的内容。在业务层的 Web 应用程序中，我们主要用到了 GET 请求、POST 请求以及 DELETE 请求。本文针对每个数据表都设计了一个资源类，分别是 PatientsResources 类、CasesResources 类、ECGResources 类、CDGResources 类以及 TimesResources 类。这些类是对外的接口，客户端通过特定的 RESTful API 定位到特定的资源类中的指定函数，然后在函数中处理具体的业务逻辑，如数据存储、数据查询、数据分析等。表 5-5 所示为常用的 RESTful API 及其描述。

表 5-5 常用 RESTful API

API	HTTP 方法	参数	描述
/patients	GET	无	获取所有病人的基本信息
/patients/id	GET	病人 id	根据 id 获取病人的基本信息
/patients/adnum	GET	住院号	根据病人住院号获取病人基本信息
/patients/insert	POST	基本信息	向 patients 表中插入病人基本信息
/patients/delete/id	DELETE	病人 id	根据 id 从 patients 表中删除指定病人的信息

/cases/patientId	GET	病人 id	根据 id 获取病人病历信息
/cases/insert	POST	病历信息	向 cases 表插入病人病历信息
/cases/delete/patientId	DELETE	病人 id	根据 id 从 cases 表中删除指定病人的病历信息
/ecg/testId	GET	测试号	根据 ECG 测试号获取指定的原始心电数据
/ecg/insert	POST	心电数据	向 ecg 表中插入心电数据
/ecg/delete/testId	DELETE	测试号	根据测试号从 ecg 表中删除指定 ECG 数据
/ecg/upload	POST	心电数据	向服务器上次心电数据文件
/ecg/learn	GET	目录路径	对指定路径中的心电数据进行分析诊断
/cdg/testId	GET	测试号	查询指定测试号的 CDG
/cdg/insert	POST	CDG 数据	向 cdg 表中插入 CDG 数据
/cdg/delete/testId	DELETE	测试号	根据测试号从 cdg 中删除指定的 CDG 数据

5.3.3 负载均衡配置

在多台服务器上部署 Web 应用程序时，有必要对集群进行负载均衡的配置。本文选用 Nginx 来作为反向代理服务器，从而实现对业务层的负载均衡。由前文所述可知，MySQL 集群采用的是一主多备的水平扩展方案，这样做的一个好处就是可以实现读写分离：让主数据库处理写请求，然后将更新结果同步到备数据库中；让备数据库处理只读请求，由于没有对数据库中的数据进行更改，因此无需进行数据同步。因此，本文拟采用 Nginx 对业务层进行读写分离以及负载均衡。如图 5-7 所示为业务层负载均衡的方案图，本文选取配置较高的 master 服务器作为写服务器，专门响应写请求。剩余的三台 slave 服务器作为只读服务器，并对客户端的读请求进行负载均衡。

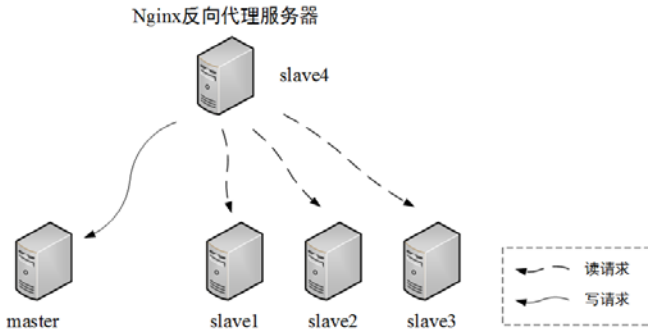

图 5-6 Nginx 负载均衡与读写分离

本文将在 Storm 集群服务器之外的 slave4 服务器上部署 Nginx 反向代理服务器，部署步骤如下：

（1）安装 PCRE 库。安装 PCRE 库的目的是使 Nginx 支持 Rewrite 功能，从官网下载最新的安装包并解压，使用 make 命令进行编译安装即可。

（2）安装 Nginx。从官网下载最新的 Nginx 安装包并解压，使用 make 命令进行编译安装即可。

（3）配置 Nginx。Nginx 的配置文件位于目录/usr/local/nginx/conf/nginx.conf 之下，通常包含[server]、[location]、[upstream]等项。通过配置[server]项可以实现 Nginx 根据需要将不同的请求转发到不同的服务器上。每一个 URL 请求都会对应一个服务，Nginx 可以通过配置[location]项来实现对于不同的 URL 路径将请求转发到指定的服务器上。而[upstream]项用来对负载均衡进行配置，它指定一组服务器，并将接受到的请求安装预先配置的负载均衡策略分发到这些服务器上。Nginx 配置文件的主要内容如下所示：

```
server {
  listen        8080
  server_name   slave4
  location / {
    proxy_pass  slave4;
    //如果是写请求，则转发到 master 上
    if($request_method="POST" || $request_method="DELETE"){
        proxy_pass master;
    }
  }
}
upstream slave4 {
  //对 ip 进行哈希的负载均衡策略
  ip_hash;
  //如果是读请求，则根据负载均衡策略转发到以下服务器上
  server slave1:8080;
  server slave2:8080;
  server slave3:8080;
}
```

5.4 表现层的实现

平台采用的是 B/S 架构，因此，表现层是通过 Web 浏览器展示的。表现层的实现用到了 Bootstrap、JQuery、plotly.js、DataTable 等框架，其中 Bootstrap 框架用于对基础页面布局的设计；JQuery 则用于编写页面的特效，并结合 Ajax 代码与后台的 Web 应用程序进行数据交互；plotly.js 用于将对心电数据的学习训练结果绘制成 CDG 环；DataTable 将病人基本信息、病历信息、心肌缺血诊断结果等以表格的形式展现出来，使得用户能够对数据一目了然，以便对数据执行便捷的搜索查询。

如图 5-7 所示为表现层界面的结构图，由图可知，表现层一个分为了七大模块，分

别是首页、关于、病历查询、病历录入、CDG 诊断、联系我们和友情链接。其中，病历查询界面是查询病人基本信息、病历信息的主要界面；病历录入则是录入病人信息的界面，又可以分为病历添加和病历修改；CDG 诊断是上传 ECG 数据并对数据进行早期心肌缺血诊断的界面。

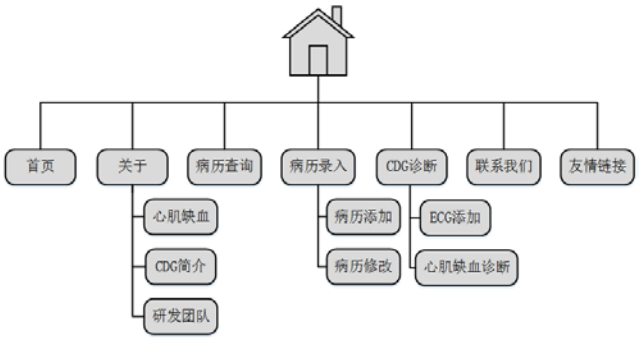

图 5-8 表现层界面结构图

5.4.1 病历查询模块

病历查询模块用于根据用户输入的筛选条件，对病人的基本信息、病历信息、CDG 诊断结果进行查询。模块利用 DataTable 框架将这些信息显示成表格的形式，并提供了便捷的数据搜索功能。另外，模块利用了 JavaScript 的 JQuery 库对页面进行了特效渲染，是界面操作更具友好性。模块对三维 CDG 环的绘制采用的是 plotly.js 库，该库提供了强大的图形绘制功能，并带有图形保存的功能，有利于对 CDG 的进一步的研究分析。病历查询模块界面如图 5-9 所示。

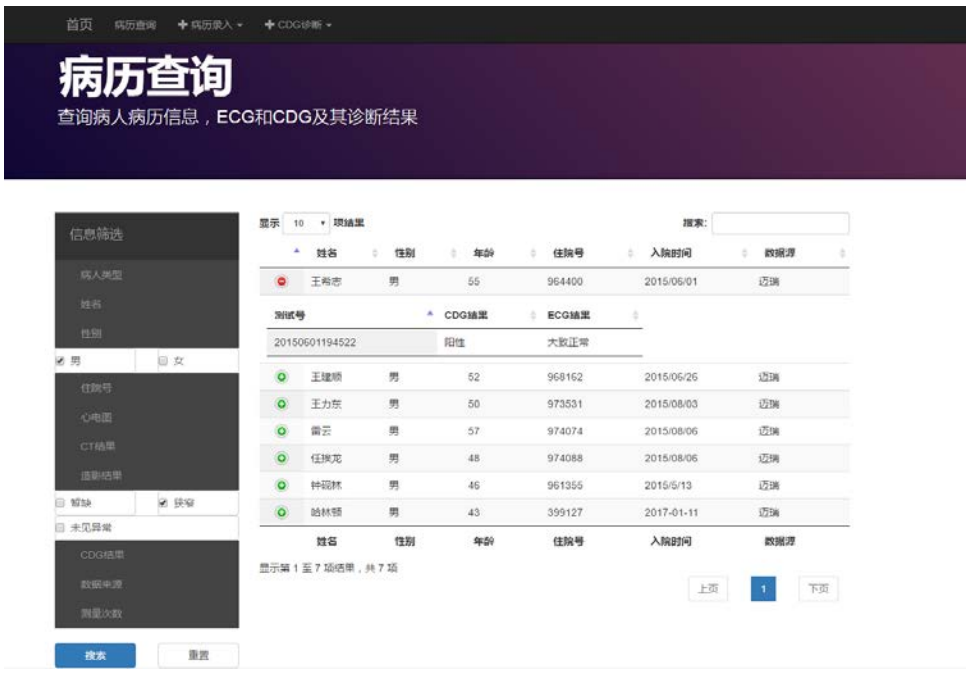

图 5-9 病历查询模块界面

在进行病历查询操作时，首先用户需要在界面左侧的信息筛选栏目勾选特定的筛选信息，用鼠标点击搜索按钮后，符合条件的病人的基本信息就会以表格的形式展现在界面右侧。另外，点击表格中病人所在行就可以查看该病人的病历信息，如图 5-10 (a)所示。用户只需点击病人所在行左边的加号符号，即可查看该病人对应的 ECG 诊断结果以及 CDG 诊断结果，再次点击 CDG 诊断结果单元就可以查看病人的 CDG 环，如图 5-10 (b)所示。

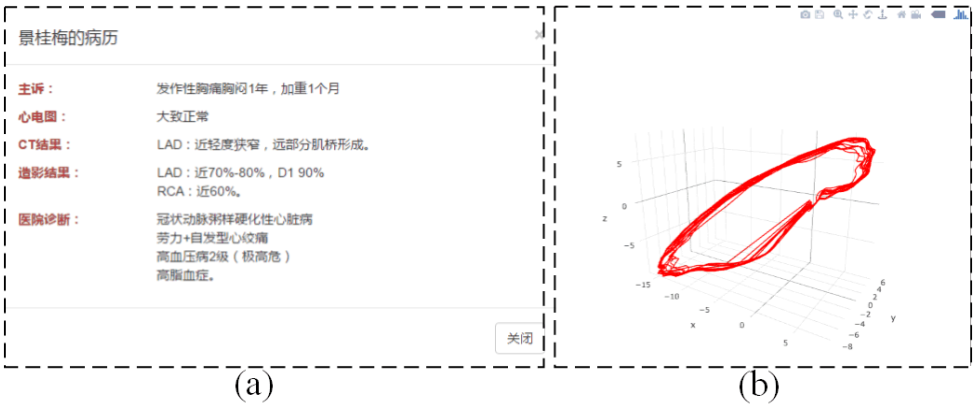

图 5-10 病历信息与 CDG 环显示

鼠标在搜索按钮、加号符号等组件上的每一次点击都会触发表现层后台的JavaScript回调函数，回调函数通过 Ajax 异步技术向业务层获取数据，进而在 Web 浏览器上展现出来。病历查询模块的序列图如图 5-11 所示。

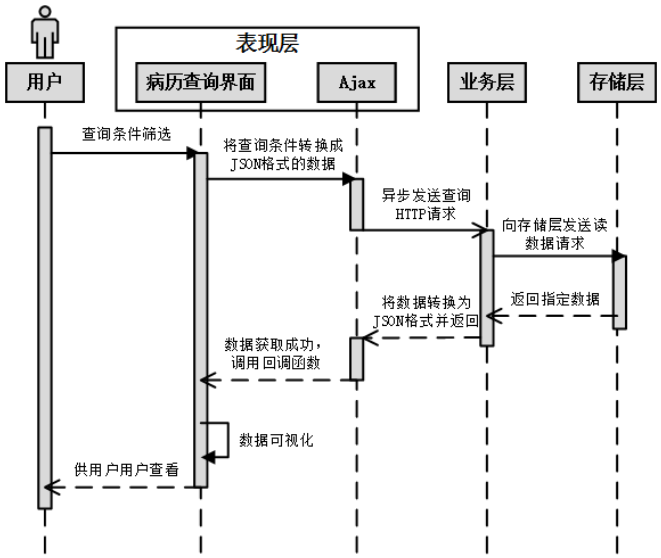

图 5-11 病历查询模块序列图

5.4.2 病历录入模块

病历录入模块包含两个部分，分别是病历添加和病历修改。

病历添加是将病人的基本信息（姓名、性别、年龄、住院号等）和病历信息（主诉、入院日期、入院初次诊断、心电图结果、造影结果等）发送到存储层进行持久性存储。病历添加的界面如图 5-12 所示。用户需要现在图 5-12 表单上填写病人的信息，然后点击添加按钮即可触发后台的 JavaScript 代码，通过 Ajax 技术将数据发送到业务层，再由业务层传输到存储层进行持久性存储。病历添加的序列图如图 5-13 所示。

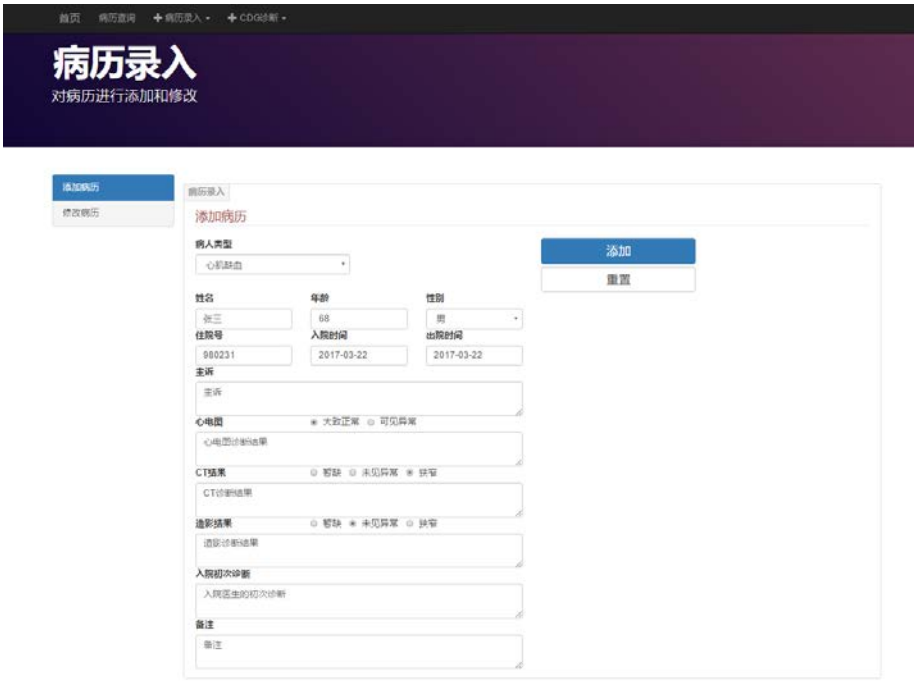

图 5-12 病历添加界面

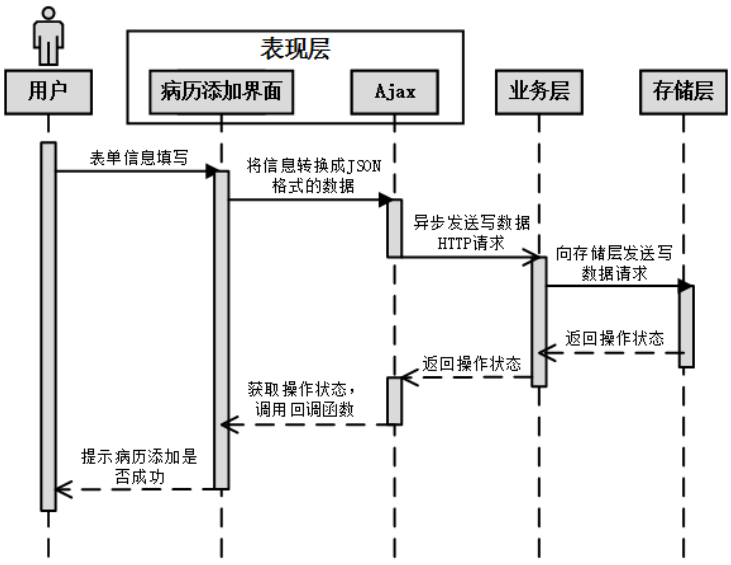

图 5-13 病历添加序列图

病历修改是对已经在存储在存储层的病历信息进行修改。通常情况下，病人的病历信息会随着每一次诊断而有变动，可以是修改原有的诊断结果，也可以是添加新的诊断结果。因此，平台有必要为用户提供病历修改的功能。病历修改模块的界面如图 5-14

所示。首先，用户必须根据病人的住院号进行病历搜索，如果该病人已经存有病历信息就会有病历修改窗口弹出，然后在该窗口对病历进行修改，最后点击保存修改即可触发后台 JavaScript 代码，通过 Ajax 技术与业务层进行数据交互。病历修改的序列图如图 5-15 所示。

图 5-14 病历修改界面

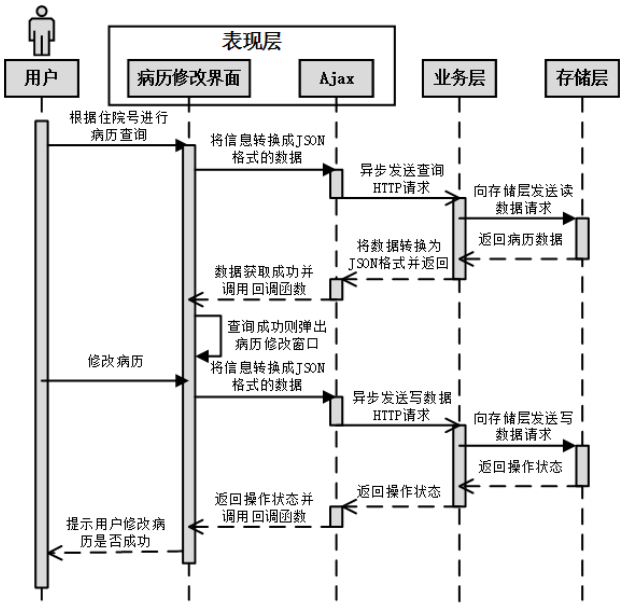

图 5-15 病历修改序列图

5.4.3 CDG 诊断模块

CDG 诊断模块用于对用户上传的心电数据进行心肌缺血诊断，是表现层的核心模块。首先，用户需要将心电数据文件上传到服务器上，其中心电数据为文本文件，文件

名的格式为[住院号_测试号]。另外，可以将多个心电数据打包成一个 zip 压缩包，再将压缩包上传到服务器，由业务层对压缩包进行解压，然后放到 Redis 消息中间件中以供计算层进行处理。CDG 诊断模块的界面如图 5-16 所示。在上传数据的过程中，进度条会将上传进度信息反馈给用户，提高了用户与平台的交互性。上传完成后，进度条下面会出现上传状态（上传成功后上传失败）来提示用户进行下一步的操作。心电数据上传成功后就可以点击 CDG 诊断按钮进行 CDG 诊断。与上传心电数据类似，等待 CDG 诊断完成后，界面会提示诊断状态（诊断成功或诊断失败），以此提示用户进行下一步操作。CDG 诊断模块的序列图如图 5-17 所示。

图 5-16 CDG 诊断模块界面

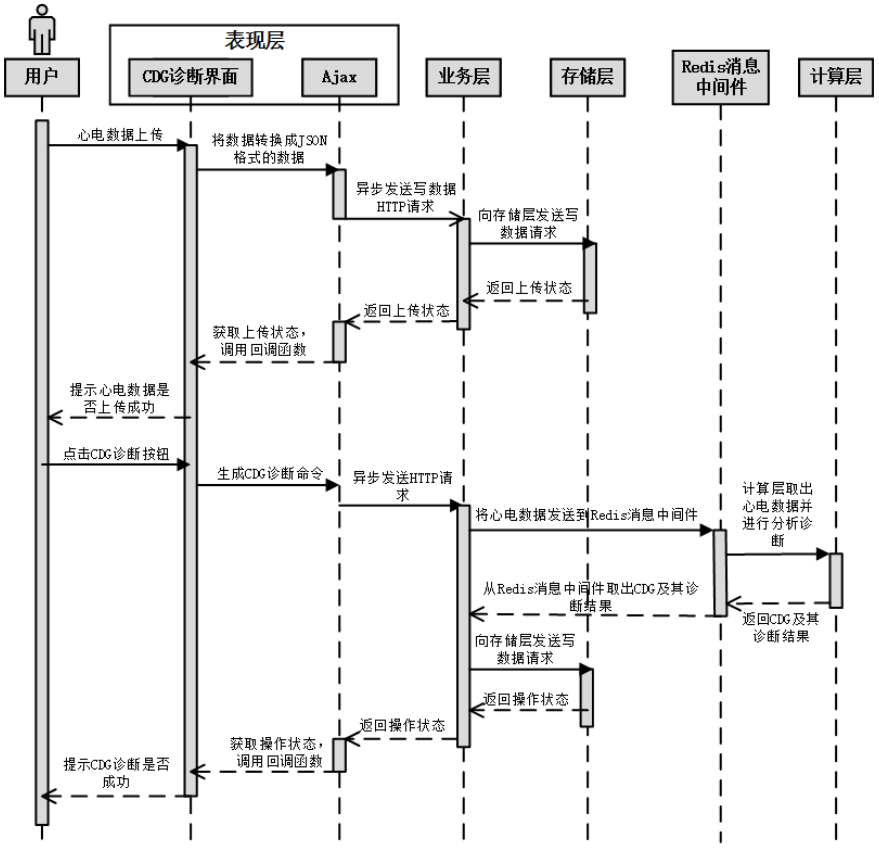

图 5-17 CDG 诊断模块序列图

5.5 本章小结

本章实现了平台的存储层、业务层和表现层，结合上一章节实现的计算层，构成了一个完整的心肌缺血早期诊断平台，为用户提供高效、可靠的心肌缺血早期诊断服务。首先，本文介绍了存储层的实现。存储层基于 MySQL 数据库，根据具体的需求，一共设计了五个数据表（patients、cases、ecg、cdg 和 times）和两个存储过程（patientInfoQuery 和 cdgQuery）。为了提高存储层的可用性，本文对 MySQL 数据库采取了一主多备的水平扩展策略。然后，本文对业务层的实现进行了详细的描述。业务层由多个 Web 应用程序组成的集群组成，并在 Web 应用程序之上搭建 Nginx 反向代理服务器达到负载均衡和读写分离的效果。Web 应

用程序是基于 Dropwizard 框架开发的 RESTful 风格的应用，它对外提供 RESTful API 来响应业务请求。最后，本章分析了表现层的实现，表现层基于 Bootstrap、JQuery、plotly.js 等框架开发，并通过 Web 浏览器展现。表现层主要的三大模块是病历查询模块、病历录入模块和 CDG 诊断模块，本章不仅介绍了它们的界面设计，而且还对其背后的工作流程以序列图的形式展现出来。至此，心肌缺血早期诊断平台的四大层次已经完整地组合在一起，进而可以为用户提供心肌缺血早期诊断的服务。

结论与展望

在众多心血管疾病中，心肌缺血是一种常见的病态，已经严重威胁了国民的生命健康。利用确定学习理论，可以实现对标准十二导联心电图进行早期的心肌缺血诊断。本文搭建了一个基于 Storm 实时流计算框架的心肌缺血早期诊断平台，平台采用 B/S 架构开发，结合互联网上最新的开源成果，实现了数据分析计算、数据存储和信息管理的功能，为用户提供了高效、可靠的心肌缺血诊断服务。

本文主要做了如下工作：

（1）对本文所述平台进行了系统的需求分析，得出平台主要包含三大功能，分别是分析计算功能、数据存储功能和信息管理功能。然后针对每一个功能提出了相对应的解决方案，最后得出整个平台的软件架构，该架构将系统划分为计算层、存储层、业务层和表现层四大层次，使得可以按照层次来对平台进行实现。

（2）将已有的 C++版心肌缺血诊断程序移植到 Storm 框架上，实现了计算层。首先，根据 Storm 流计算框架的特点，本文将已有的诊断程序分割为三大模块，分别是预处理模块、ST-T 段截取模块以及学习训练，并将每一个模块实现为 Storm 中的一个 Bolt。最后本文给出了整个诊断程序的拓扑结构，整个拓扑结构一共有五类节点，读取输入数据的 Read Redis Spout 节点、对数据进行预处理的 Pretreat Bolt 节点、进行 ST-T 段截取的 CutST Bolt 节点、实现确定学习算法的 Learn Bolt 节点与输出数据的 Write Redis Bolt 节点。此外，本文采用 Redis 消息中间件作为计算层与业务层之间数据交互的媒介，使得它们之间的耦合性变低。

（3）实现了存储层、业务层和表现层，与计算层组合起来，构成了一个完整的软件系统。存储层的实现基于 MySQL 关系型数据库，为了提高系统的可用性，本文采用了一主多备的水平扩展策略搭建了 MySQL 集群。在业务层的实现上，本文基于 Dropwizard 框架开发了具有 RESTful 风格的 Web 应用程序，通过对外提供 RESTful API 来响应各种业务请求。为了提高业务层的效率，本文还搭建了 Nginx 反向代理服务器对业务层进行负载均衡以及读写分离。最后，本文基于 Bootstrap、JQuery、plotly.js、DataTable 等框架设计了操作便捷、简洁美观的表现层，使得用户可以简单方便地使用平台的各种服务。

以上三点是本文的主要研究成果，该平台基本上满足了用户通过 Web 浏览器进行远程的实时心肌缺血诊断和病人信息管理的需求，但是它仍然有一些需要进一步改进的

地方，主要有以下几点：

（1）缺乏批量计算的功能。目前计算层是基于 Storm 流计算框架进行开发的，该框架主要应用于实时计算领域，在批量计算的应用场景效率不高。而本人所在实验室已经开发出基于 Hadoop 框架的心肌缺血诊断程序，其在批量计算上有着较高的性能。因此，将来可以考虑将 Hadoop 与 Storm 结合起来，使得平台既可以提供实时计算的服务，又可以提供批量计算的服务。

（2）功能不够完善。目前，平台的功能并不是很完善，例如在病历管理方面，本平台只能查询到心电图的诊断结果，尚未支持心电图数据的查看。另外，在 CDG 诊断方面，系统只支持对已经录入病历病人的 ECG 进行诊断，将来可以考虑将病历录入与 CDG 诊断分离。

（3）安全性不足。本平台尚未完成对权限管理模块的开发，因此存在安全性的问题。因此，将来需要为平台添加权限管理功能，对不同类型的用户设立不同的使用权限。

参考文献

[1]陈伟伟,高润霖,刘力生,朱曼璐,王文,王拥军,吴兆苏,李惠君,顾东风,杨跃进,郑哲,蒋立新,胡盛寿. 《中国心血管病报告 2015》概要[J]. 中国循环杂志,2016,06:521-528.

[2]记者 王平 通讯员 尹沅沅 邢永田. 心血管病成健康“第一杀手”[N]. 河南日报,2016-11-14003.

[3]朱明星,李北方,刘仁光. 常规心电图——立体心电图应用研究现状[J]. 心血管病学进展,2010,04:612-616.

[4]杜鹏飞. 心电信号的预处理及特征提取算法研究[D].郑州大学,2015.

[5]季虎. 心电信号自动分析关键技术研究[D].国防科学技术大学,2006.

[6]孟欢欢. 心电信号自动分析的几种算法研究[D].清华大学,2014.

[7] Pan J, Tompkins W J. A real-time QRS detection algorithm. Biomedical Engineering, IEEE Transactions on, 1985 (3): 230-236.

[8] Chen H C, Chen S W. A moving average based filtering system with its application to real-time QRS detection [C]. Computers in Cardiology, 2003. IEEE, 2003: 585-588.

[9] Tarassenko L, Clifford G, Townsend N. Detection of ectopic beats in the electrocardiogram using an auto-associative neural network. Neural Processing Letters, 2001, 14(1): 15-25.

[10] Mehta S S, Lingayat N S. Development of SVM based classification techniques for the delineation of wave components in 12-lead electrocardiogram. Biomedical Signal Processing and Control, 2008, 3(4): 341-349.

[11]司徒志强,王明飞,张晶晶. 心电图 ST-T 改变对冠心病的诊断价值[J]. 当代医学,2011,(17):88-89.

[12]Minchole A, Skarp B, Jager F, et al. Evaluation of a root mean squared based ischemia detector on the long-term ST database with body position change cancelation. Comput Cardiol, 2005, 32: 853–856

[13]Stadler R, Lu S, Nelson S, et al. A real-time ST-segment monitoring algorithmfor implantable devices. J Electrocardiol, 2011, 34: 119–126

[14]Garcia J, Sornmo L, Olmos S, et al. Automatic detection of ST-T complex changes on the ECG using filtered RMS difference series: application to ambulatory ischemia monitoring. IEEE Trans Biomed Eng, 2000, 47: 1195–1201

[15]Smrdel A, Jager F. Automated detection of transient ST-segment episodes in 24h electrocardiograms. Med Biol Eng Comput, 2004, 42: 303–311

[16] Maglaveras N, Stamkopoulos T, Pappas C, et al. An adaptive backpropagation neural

network for real-time ischemia episodes detection: development and performance analysis using the European ST-T database. IEEE Trans Biomed Eng, 1998, 45: 805–813

[17] Papaloukas C, Fotiadis D I, Likas A, et al. An ischemia detection method based on artificial neural networks. Artif Intell Med, 2002, 24: 167–178

[18] Afsar F A, Arif M, Yang J. Detection of ST segment deviation episodes in ECG using KLT with an ensemble neural classifier.

[19] Exarchos T P, TsipourasM G, Exarchos C P, et al. A methodology for the automated creation of fuzzy expert systems for ischaemic and arrhythmic beat classification based on a set of rules obtained by a decision tree. Artif Intell Med, 2007, 40: 187–200

[20] Cong WANG, Xunde DONG. A new method for early detection of myocardial ischemi-a:cardiodynamicsgram(CDG)[J]. Science China(Information Sciences), 2016, 01: 95-105.

[21]姚成. 心电信号智能分析关键技术研究[D].吉林大学,2012.

[22]Tom White. Hadoop: The definitive guide[M]. 4th edition. United States: O'Reilly Media, Inc. 2012.

[23]Sanjay Ghemawat, Howard Gobioff, Shun-Tak Leung. The Google File System[J]. ACM SIGOPS operating systems review, 2003, 37(5): 29-43.

[24]Chang F., Dean J., Ghemawat S., et al. Bigtable: A Distributed Storage System for Structured Data[J]. OSDI'06: Seventh Symposium on Operating System Design and Implementation, Seattle, WA, November, 2006. 2006. p. 205--218.

[25] Jason Baker, Chris Bond, James C Corbett, et al. Megastore: Providing Scalable, Highly Available Storage for Interactive Services[J]. Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 9-12, 2011

[26]Spanner

[27] Jeffrey Dean, Sanjay Ghemawat. MapReduce: Simplified Data Processing on Large Clus- ters[J]. Communications of the ACM, 2008, 51(1): 107-113.

[28]罗旭,刘友江. 医疗大数据研究现状及其临床应用[J]. 医学信息学杂志,2015,(05):10-14.

[29]高汉松,肖凌,许德玮,桑梓勤. 基于云计算的医疗大数据挖掘平台[J]. 医学信息学杂志,2013,(05):7-12.

[30]徐凯田. 基于大数据的智慧移动医疗信息系统结构研究[D].青岛科技大学,2015.

[31]袁胜. 基于 Hadoop 的心肌缺血辅助诊断工作站的设计及实现[D].华南理工大学,2016.

[32]周仁义. 心电信号处理关键技术的研究与实现[D].东北大学,2014.

[33]卢志强,张艳军,崔广智,庄朋伟,张金保. 心肌缺血模型的制作方法研究进展[J]. 中国药理学通报,2012,(08):1053-1057.

[34] Roger V.L., Go A.S., Lloyd-Jones D.M., et al. Heart disease and stroke statistics—2011 update a report from the American Heart Association[J]. Circulation, 2011, 123(4): 18-209

[35]朱明星,李北方,刘仁光. 常规心电图——立体心电图应用研究现状[J]. 心血管病学进展,2010,(04):612-616.

[36]陈清启. 心电向量图在心肌梗死诊断和鉴别诊断中的优势[J]. 江苏实用心电学杂志,2013,(03):617-624.

[37] 毛玲,张国敏.心电图 ST 段形态分析方法研究[J].信号处理, 2009, 09: 1360-1365.

[38] Zhao Shen, Chao Hu. An algorithm of ST segment classification and detection[A]. IEEE International Conference on Automation and Logistics[C]. New York: IEEE, 2010: 559- 564.

[39] Daniel Lemire, Chantal Pharand.Wavelet time entropy T wave morphology and myocardial ischemia[J]. IEEE Trans Biomed Eng, 2000, 47(7): 967-970.

[40]任莉娜,齐国先. T 波电交替的研究进展[J]. 心血管病学进展,2011,(01):107-111.

[41] Bruce D. Nearing, Richard L. Verrier. Modified moving average analysis of T-wave alternans to predict ventricular fibrillation with high accuracy. Journal of Applied Physiology[J] Feb 2002, 92 (2) 541-549

[42]田景坤. 基于确定学习的心肌缺血早期检测技术研究及 C++实现[D].华南理工大学,2016.

[43] Cong Wang, David J.Hill. Learning From Neural Control[J]. IEEE Trans Neural Netw, 2006, 17(1): 130-146.

[44] Wang C, Hill D J. Deterministic learning theory for identification, recognition, and con-trol[M]. United States: CRC Press, 2009.

[45]吴玉香,王聪. 基于确定学习的机器人任务空间自适应神经网络控制[J]. 自动化学报,2013,06:806-815.

[46] 曾玮.基于确定学习理论的人体步态识别研究[D].广州：华南理工大学, 2012.

[47] 文彬鹤.基于确定学习理论的轴流压气机旋转失速建模与检测[D].广州：华南理工大学, 2013.

[48] 陈填锐,确定学习理论与智能振动故障诊断[D].广州：华南理工大学, 2010.

[49] 刘明星.心肌缺血检测的实用系统开发[D].广州：华南理工大学, 2015.

[50] Kurdila A.J., Narcowich F.J., Ward J.D. Persistancy of excitation in identification using

radial basis function approximants[J]. SIAM J. Control and Optimization, 1995, 33(2): 625-642.

[51] Liu Tengfei, Cong Wang. Deterministic learning and rapid dynamical pattern recognition of discrete-time systems[A]. IEEE International Symposium on Intelligent Control[C]. New York: IEEE, 2008: 1091-1096.

[52]文斯民. 基于确定学习的心肌缺血早期诊断系统的配套信息管理系统的开发[D].华南理工大学,2016.

[53]靳永超,吴怀谷. 基于 Storm 和 Hadoop 的大数据处理架构的研究[J]. 现代计算机(专业版),2015,04:9-12.

[54]张俊林. 大数据日知录: 架构与算法[M]. 第 1 版. 北京: 电子工业出版社, 2014.

[55]R. Ruan, M. Deng and C. Wang, "Implementation of a flexible and extensible clinical data management system for cardiovascular disease," 2016 35th Chinese Control Conference (CCC), Chengdu, 2016, pp. 9394-9399.

[56] Alexandros Dallas. RESTful Web Services with Dropwizard[M]. Packt Publishing, 2014.

[57] 邱祝文. 基于 redis 的分布式缓存系统架构研究[J]. 网络安全技术与应用,2014,10:52+54.

[58]王利萍. 基于 Nginx 服务器集群负载均衡技术的研究与改进[D].山东大学,2015.

[59]屈国庆. 基于 Storm 的实时日志分析系统的设计与实现[D].南京大学,2016.

攻读硕士学位期间取得的研究成果

一、已发表（包括已接受待发表）的论文，以及已投稿、或已成文打算投稿、或拟成文投稿的论文情况**（只填写与学位论文内容相关的部分）：**

序号	作者（全体作者，按顺序排列）	题　　目	发表或投稿刊物名称、级别	发表的卷期、年月、页码	相当于学位论文的哪一部分（章、节）	被索引收录情况
1	Ruan Runxue; Deng Muqing; Wang Cong	Implementation of a flexible and extensible clinical data management system for cardiovascular disease	Proceedings of the 35th Chinese Control Conference, CCC 2016	2016-August, pp.9394-9399	第五章	已被 EI 索引