分类号_____ 密级 _____

UDC _____

学 位 论 文

中厚板轧机二级通信平台调度监控系统的设计与实现

作 者 姓 名： 任晓东

指 导 教 师： 常桂然 教授

东北大学计算中心

申请学位级别： 硕士 学 科 类 别： 工学

学科专业名称： 计算机应用技术

论文提交日期： 2011 年 5 月 7 日 论文答辩日期： 2011 年 6 月 19 日

学位授予日期： 答辩委员会主席： 陶振凯 高工

评 阅 人： 易秀双 副教授 孙建伟 研究员

东 北 大 学

2011 年 6 月

A Thesis for the Degree of Master in Computer Application Technology

Design and Implementation of a Dispatching and Monitoring System for the Medium Plate Mill Level 2 Communication Platform

By Ren Xiaodong

Supervisor: Professor Chang Guiran

Northeastern University
June 2011

独创性声明

本人声明，所呈交的学位论文是在导师的指导下完成的。论文中取得的研究成果除加以标注和致谢的地方外，不包含其他人已经发表或撰写过的研究成果，也不包括本人为获得其他学位而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：任晓东，

日　　期：2011.6.8

学位论文版权使用授权书

本学位论文作者和指导教师完全了解东北大学有关保留、使用学位论文的规定：即学校有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人同意东北大学可以将学位论文的全部或部分内容编入有关数据库进行检索、交流。

作者和导师同意网上交流的时间为作者获得学位后：

半年 ☑　一年□　一年半□　两年□

学位论文作者签名：任晓东，　导师签名：

签字日期：2011.6.8　签字日期：2011.6.8

中厚板轧机二级通信平台调度监控系统的设计与实现

摘 要

中厚板是指厚度在 4.5mm-100.0mm 的钢板，主要应用于机械制造、建筑工程、桥梁建造等诸多领域，是基础设施建设和重型机械制造的重要原材料。轧机作为中厚板生产线的核心，在中厚板的生产中有着举足轻重的地位，所以目前国内外很多钢铁公司都逐步地尝试使用计算机控制系统来辅助传统的工艺生产，这不仅大幅度提高了生产效率，更使得生产的精确度和可靠度得到了保障。然而国内多数钢铁公司所使用的计算机控制系统的关键技术来源于国外，造成核心技术垄断、后期维护成本过高等问题，因此，开发一个拥有自主知识产权的、跨平台的、通用的轧机二级通信调度监控系统，具有深远的市场意义和社会意义。

公共对象请求代理体系结构（CORBA）作为一种中间件，提供了分布式异构对象的互操作，它定义的对象请求代理（ORB）保证了对象在异构环境间的通讯和交流。自适应通信环境（ACE）是一种开源的、免费的面向对象组件和工具包，它为通信软件提供了核心并发和分布机制。 TAO 是一个免费开源的实时 CORBA 平台的实现，它具有高效的、可预知的、可升级的端到端传输特性。

本文以首钢 3500mm 中厚板生产线为研究背景，以国家重点实验室开放课题《中厚板轧机二级系统开发》完成的原型系统为研究基础，对 CORBA、ACE、TAO 通信中间件和 OO4O、数据仓库（DW）、联机分析处理（OLAP）等关键技术进行了进一步的研究，提出了基于 ACE+TAO+OO4O 通信框架和基于 DW+OLAP 分析决策框架的中厚板轧机二级通信平台调度监控系统。本文对系统进行了概要设计和详细设计，进一步分析了多进程间的通信关系，重点讨论了数据库的访问方式和数据流向等问题，同时对监控子系统和自学习子系统进行了详细设计。最终完成了中厚板轧机二级通信平台调度监控系统的实现。

将实现的系统在轧机 PLC 平台下进行系统功能验证，测试的结果证明了本文提出的基于 ACE+TAO+OO4O 通信框架和基于 DW+OLAP 分析决策框架的中厚板轧机二级通信平台调度监控系统设计方案的可行性和正确性。最后，对系统的整个设计与实现过程进行了总结，并对系统功能、扩展性能以及系统的应用领域进行了展望。

关键词：中厚板轧机；调度监控；自适应通信环境（ACE）；OO4O；联机分析处理（OLAP）

Design and Implementation of a Dispatching and Monitoring System for the Medium Plate Mill Level 2 Communication Platform

Abstract

The medium plate is a kind of steel with the thickness of 4.5mm-100.0mm, which is mainly applied to machinery manufacturing, construction engineering, bridge construction and many other fields, it is an important raw material in infrastructure construction and heavy machinery manufacturing. The mill, as the core in the production line of the medium plate, plays a vital part in the production. Therefore, recently many iron and steel companies at home and abroad are trying to use computer control system to support the traditional production process, which not only can substantially increase production efficiency, but also can make the accuracy and reliability of production be guaranteed. However, key technologies of computer control system in most of the domestic iron and steel companies are from foreign countries, which lead to some problems such as monopoly of the core technology, high cost of maintenance in later stage,etc. Hence, it has a far-reaching significance for the market and the society to develop a general, cross-platform dispatching and monitoring system for the mill level 2 communication platform with independent intellectual property rights.

CORBA, as a middleware, provides the interoperability of distributed heterogeneous object. The ORB, which is the object of what it defines, has ensured the communication and exchange of the objects among heterogeneous environment. ACE is a kind of object-oriented, open source and freely available component and toolkit, it has provided core concurrency and distribution mechanism for communication software. TAO is a concrete implementation of a free, open resource and real-time CORBA, it has the features of high-efficiency, predictability, scalability and end to end transmission.

This thesis carries out a further study about key technologies of CORBA, ACE, TAO, OO4O, DW and OLAP with the production line of the medium plate of 3500mm in Shougang as the research background. It is based on the research of prototype system which was finished in the libraries'open subject "The development for a system of the Medium Plate Mill Level 2". This thesis also proposes a dispatching and monitoring system for the medium

plate mill level 2 communication platform based on the communication framework of ACE+TAO+OO4O and the analysis and decision framework of DW+OLAP. This thesis conducts the outline design and detailed design for the system. At the same time, it analyzes the further relationship of communications among multiple processes and focuses on the problems of the database access methods and the data stream, this thesis pursues the detailed design of monitoring subsystem and self-learning subsystem, at last, it implements the dispatching and monitoring system for the medium plate mill level 2 communication platform.

After verifying the system on mills PLC platform, it has been proved the correctness and feasibility of the design plan for the dispatching and monitoring system for the medium plate mill level 2 communication platform on the base of the communication framework of ACE+TAO+OO4O and the framework of analysis and decision of DW+OLAP, which are proposed in this thesis.At last, the entire development process of this system has been summarized, meanwhile, this thesis looks forward to the function of the system, the ability of extension and the fields of applications of it.

Key words: Medium Plate Mills; Dispatching and Monitoring; Adaptive Communication Environment (ACE); OO4O; On-Line Analytical Processing(OLAP)

目 录

独创性声明 I

摘 要 II

Abstract III

第 1 章 绪 论 1

1.1 研究背景 1

1.2 国内外研究现状 3

1.3 研究目的与意义 4

1.4 本文主要工作 4

1.5 本文组织结构 5

第 2 章 基础知识和关键技术介绍 7

2.1 中间件及面向对象中间件 7

2.2 分布式中间件 CORBA 10

2.3 自适应中间件 ACE 11

2.4 The ACE ORB 13

2.5 Oracle 数据库访问技术 OO4O 14

2.6 DW+OLAP 15

2.6.1 数据仓库 DW 16

2.6.2 联机分析处理 OLAP 16

2.7 本章小结 18

第 3 章 通信调度监控系统的分析与设计 19

3.1 中厚板轧机二级通信平台调度监控系统的需求分析 19

3.2 中厚板轧机二级通信平台调度监控系统的概要设计 22

3.2.1 系统框架的设计 22

3.2.2 进程间通信方式的讨论 25

3.2.3 数据流分析 26

3.2.4 数据库访问方式的讨论 27

3.3 中厚板轧机二级通信平台调度监控系统的详细设计 28

3.3.1 轧机二级通信子系统的设计 28

3.3.2 轧机二级监控子系统的设计.....35
3.3.3 轧机二级自学习子系统的设计.....36
3.4 本章小结.....40
第 4 章 通信调度监控系统的实现.....41
4.1 开发平台的选型.....41
4.2 轧机二级通信子系统的实现.....42
4.2.1 接收进程的实现.....42
4.2.2 守候进程的实现.....46
4.2.3 发送进程的实现.....49
4.2.4 预计算进程的实现.....50
4.2.5 数据库的实现.....51
4.3 轧机二级监控子系统的实现.....53
4.3.1 进程监控的实现.....54
4.3.2 数据监控的实现.....56
4.4 轧机二级自学习子系统的实现.....56
4.4.1 数据仓库的实现.....57
4.4.2 OLAP 的实现.....59
4.5 系统环境配置.....60
4.5.1 ACE+TAO 环境配置.....60
4.5.2 OWB 环境搭建.....61
4.6 本章小结.....62
第 5 章 测 试.....63
5.1 测试环境.....63
5.2 通信功能测试.....63
5.3 监控功能测试.....65
5.4 自学习功能测试.....65
5.5 本章小结.....66
第 6 章 总结与展望.....67
6.1 总结.....67
6.2 展望.....68

参考文献.....71
致谢.....75
攻读学位期间发表的论著和科研、获奖情况.....77

第1章 绪 论

中厚板是基础设施建设和大型工程、重型机械制造的重要原材料，因其广泛的用途，在钢铁行业处于重要地位。我国中厚板生产线数量也在逐年增加，为了提高生产效率和生产水平，一方面，国内许多钢铁公司引进了国外先进的中厚板生产线，然而由于对引进技术的消化吸收不彻底，对核心技术尚未能掌握，使得后续系统升级、系统维护成本较高；另一方面，国内部分钢铁公司已经对原有的中厚板生产线进行改造，虽然有所突破，然而仍然面临生产线自动化程度低、控制精度不高等问题。因此，开发一个拥有自主知识产权的、跨平台、通用的、可扩展的中厚板轧机二级通信平台调度监控系统，本文以后简称为通信调度监控系统，具有深远的市场意义和社会意义。

1.1 研究背景

中厚板是一种宽厚比很大的扁平钢材，在实际生产线中，常常称厚度 4.5mm-25.0mm 的钢板为中板[1]，厚度 25.0mm-100.0mm 的钢板为厚板，厚度超过 100.0mm 的为特厚板，而这类钢板统称为中厚板。

由于中厚板在建筑工程、机械制造、容器制造、造船、桥梁建造等领域的广泛用途，使得中厚板的市场竞争日益激烈，因此提高中厚板钢材的生产率，保证中厚板的成材率，降低生产成本逐渐成为各个厂家竞相追求的目标[2]。近些年，国内很多钢铁公司纷纷采用了国内外先进技术和设备，实现了坯料跟踪检查、钢板形状控制、生产线温度控制等过程的计算机辅助系统的应用，然而轧机作为中厚板生产线的核心部分，它与计算机控制系统的结合使用，更是提高中厚板生产率和生产质量的关键所在。目前常规的中厚板生产线的主要设备有加热炉、除磷机、粗轧机、精轧机、快速冷却控制机和矫直机构成，如图 1.1 所示。

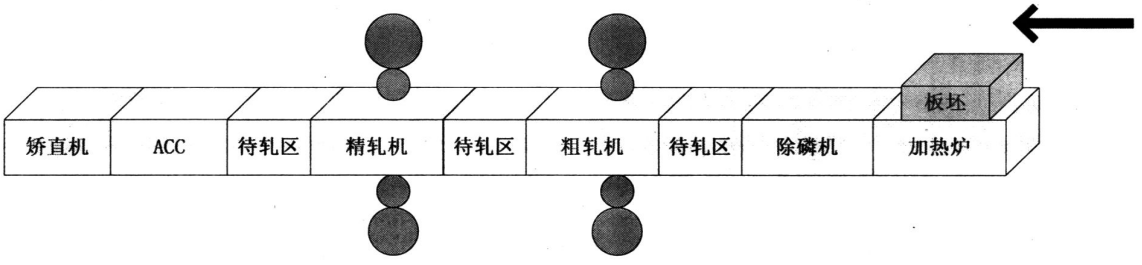

图 1.1 中厚板生产线设备布置示意图

Fig. 1.1 Layout of the mill equipments on medium plate production line

另外，生产线上还布置了相应的检测仪表，实时的检测待轧板坯的状态信息，满足轧件跟踪、自动转换等过程控制的要求。轧机两侧还分别留有一个待温轨道，利用待温轨道入口和出口处的检测仪表，可以实现钢坯的一待一轧或者多待一轧，提高轧机利用率，同时也为过程模型和自学习子系统的计算提供温度基准数据。

目前，国内现有的几家先进的中厚板计算机控制系统采用的都是分布式的计算机控制系统结构，按照功能可划分为三个层级，如图 1.2 所示。

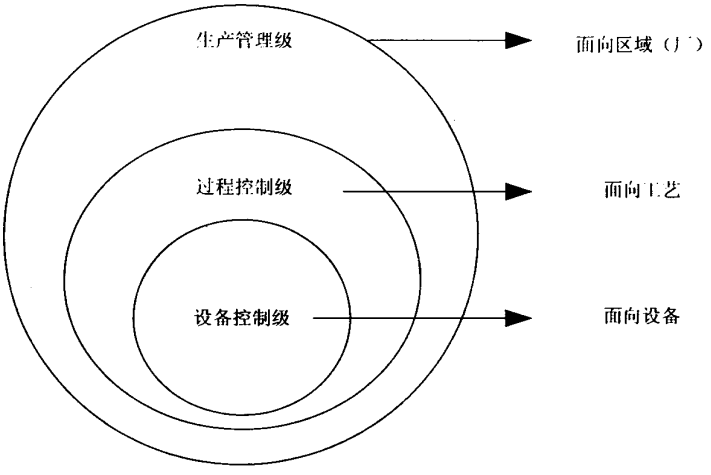

图 1.2 现代化中厚板计算机控制系统的组织结构

Fig. 1.2 The organizational structure of modern medium plate computer control system

一级为基础自动化级也称为设备控制级，它直接面向生产过程，是轧钢生产的基本环节。基础自动化级接受操作台、过程计算机、人机界面的发出的指令，来控制生产线上各个设备的稳定运行，同时将生产线上的各个检测仪表和检测设备所采集的信息实时发送给过程计算机，为之后的模型计算、修正计算和自学习子系统提供数据基础。二级为过程控制级，是轧钢生产自动化系统中重要的组成部分，它基于工业以太网和 PLC 平台进行数据通讯，利用采集数据完成任务调度、数据存储等生产工艺的要求，并根据实际生产情况进行数学模型预定值的设定。三级为生产管理级，主要控制完成从板坯入库到制成钢板成品发货的全过程，并在此基础上进行合理的生产协调与安排。

可以看出在中厚板轧机计算机控制系统中，轧机二级，既过程控制级是整个系统的核心部分，而轧机二级系统实现主要有两个难点，一是在于多进程间的通信和调度，保证轧机序列的完整性，二是对一级采样数据的存储和分析，为过程模型和自学习子系统的计算提供充分的数据基础。因此，构建一个全新的中厚板轧机二级通信平台调度监控系统的关键任务就是解决系统内多进程间通信的调度和监控问题以及实现数据的实时存储和定时分析问题。目前，国内外已有部分公司在该领域进行设计开发，其中以德国 SIEMENS 公司的轧钢自动化平台为领军水平，该平台采用基于 CORBA 中间件开发的

二级系统平台实现多进程的管理与调度以致整个生产线的全线自动化，具有良好的扩展性和可移植性等特点。另外，东北大学轧制技术及连轧自动化国家重点实验室联合东北大学计算中心也在该领域有所突破，实验室开放课题《中厚板轧机二级系统开发》已完成了中厚板轧机二级通信系统的原型[3,4]。

1.2 国内外研究现状

近些年，随着科学技术的进步和生产线环境的不断升级，中厚板生产线的自动化水平逐年提高，但是作为中厚板生产线上的核心部件——轧机，其自动化水平还有待于改善，许多中厚板生产线上的轧机操作仍然是手工完成，因此，针对中厚板轧机的计算机控制系统的研究一直成为近期轧机自动化领域的热点问题。

随着科技的进步和市场的国际化，国内很多的中厚板轧机设备的水平已得到很大幅度的提高，然而生产工艺水平不高、轧制效率不高、轧制精度低等问题日益显著。针对这些问题，国内的很多专家学者和企业技术人员提出了一些新的解决方案。牛文勇等以首钢 3500mm 中厚板轧机检测仪表、AGC 基础自动化控制系统、过程计算机系统为基础[5]，说明了轧机基础自动化系统中液压位置自动控制（HAPC）和轧制力自动控制(AFC)的原理，阐述了厚度自动控制（AGC）的控制方式以及 AGC 的运行情况。矫志杰等针对中厚板轧机，将嵌入式计算机控制系统进行改造[6]，设计了设定计算功能的在线数据流程和调用逻辑，将其划分为轧制规程计算、轧机设定及控制参数计算和模型自学习计算三个部分，发挥设定计算及生产过程中实测数据的作用。李公法等为能对轧机运行状态进行及时的远程监测与分析，采用 CORBA 技术 Web Service 技术，来解决大量数据的实时传输和防火墙穿越问题，并基于 B/S 模式[7]，以完成信号采集、实时数据动态显示功能，最终提出一种基于 CORBA 和网络服务的远程轧机状态监测系统。

国外的学者和专家近些年在轧机系统研究工作上也有卓越的成就。例如，Andreas Kugi 等考虑了液压服务器的影响，建立了液压压下系统模型，但该模型知识将轧制力昨晚一个外力来考虑，没有考虑轧制过程的影响[8]。E.Scholtz 等针对 Steckel 热轧机建立了机座模型和轧制过程模型，考虑液压系统的影响，对带材的凸度和张力动态行为进行了模拟[9]。Eugenio Brusaa 和 Luca Lemmab 针对多辊轧机辊系，基于多体动力学方法，提出了一种辊系动态计算和分析方法，可用于抑制带材产生的缺陷和预测轧辊和轴承寿命[10]。

综合起来看，到目前为止，国内外对中厚板轧机自动控制系统的研究工作一是集中于工艺模型的优化，二是集中在对个别重要设备的局部改造。这样，在一定程度上会提

高轧机生产线上的局部自动化水平，但不能实现整个轧机控制平台的集中监控，没有实现全面的自动化生产过程。

本课题充分利用东北大学轧制技术及连轧自动化国家重点实验室和首钢研究院在中厚板轧机自动化领域研究的优势，同时参考德国 SIEMENS 公司开发的基于 CORBA 的中厚板轧机计算机控制系统，提出了这种中厚板轧机二级通信平台调度监控的系统，为实现对整个轧机生产线上各个设备间通信的调度和监控提供技术支持。另外，本系统的应用将会打破西门子在此领域上的垄断地位，而且系统后期的升级改造和功能扩展也十分便利，具有强大的市场意义和社会意义。

1.3 研究目的与意义

自动化控制是衡量中厚板轧机生产线水平的重要标准，随着轧钢自动化进程的加快，实现生产线自动化生产、提高控制精度已成为人们的迫切需要，然而自动化程度低、人工操作过多仍是国内钢铁行业的瓶颈，即使某些大型企业拥有部分自动化计算机控制系统，也是从国外引进的，这不仅提高了企业运作成本，更重要的是引进而来的产品常常垄断关键技术，对后期的产品升级和功能扩展极为不便。因此，构建一个拥有自主知识产权的、跨平台、通用的、可扩展的中厚板轧机二级通信平台调度监控系统，具有深远的市场意义和社会意义。

东北大学轧制技术及连轧自动化国家重点实验室多年来在中厚板轧机计算机控制工业化应用方面积累了大量的技术成果和经验。本课题得到东北大学轧制技术及连轧自动化国家重点实验室开放课题《中厚板轧机二级系统开发》基金资助，本课题旨在利用东北大学轧制技术及连轧自动化国家重点实验室在轧钢学科和中厚板业务上的优势和东北大学计算中心计算机学科的优势构建一个具有自主知识产权的通信调度监控系统。本文是在课题前期完成的轧机二级通信系统的设计与原型实现基础上进行了进一步的开发工作，对前期完成的原型系统进行增量开发，完善各个进程间的通信关系并根据实际需要扩展相应的功能模块，完成整个二级通信系统的构建和整个轧机二级通信平台的调度与监控。最后，通过对通信调度监控系统的结合测试和现场测试，根据测试结果进行最后的调整与完善。

1.4 本文主要工作

本文主要完成了以下工作：

（1）根据对首钢 3500mm 中厚板生产线的实际考察和性能分析，在课题前期完成

的原型系统基础上，提出中厚板轧机二级通信平台调度监控系统，并对拟构建的系统进行了详细的需求分析。

（2）根据中厚板生产线的实际需求，调查研究通信调度监控系统所需的关键技术，基于 ACE、TAO 中间件技术和 OO4O、DW、OLAP 数据库技术，设计实现一个能够满足中厚板轧机需求的并且稳定的、具有自主知识产权的通信调度监控系统。

（3）最后，在东北大学轧制技术及连轧自动化国家重点实验室 PLC 测试平台下完成了系统的测试，验证了本文提出的通信调度监控系统设计方案的正确性和有效性。

在通信调度监控系统的设计和实现过程中，主要研究和解决了以下的核心技术问题：

（1）根据系统的实时性和分布式的特点，对系统中各个进程的通信关系建模，确定数据的流向，使得模型具有明确的调度关系，同时为模型预留可扩展性接口，以便后续系统的功能扩展。

（2）系统具有实时调度，快速响应的特点，同时又能够对数据进行永久性储存和分析，如何做到能实时调度又能准确的进行数据存储，是系统要解决的一个关键问题。另外，如何对已存储的历史数据进行分析处理，从而为过程模型和自学习子系统的计算提供数据基础，也是一个值得重视的问题。

（3）如何实时的反应系统中数据的流向，对系统中各个进程能够方便的控制是监控系统的要解决的问题，由于系统中各个进程通信关系的复杂性，正确分析研究每一对进程之间的通信关系是开发整个系统的关键。

1.5 本文组织结构

本文以课题前期完成的轧机二级通信系统的原型系统为基础，对各种关键技术进行了深入的分析研究，提出了中厚板轧机二级通信平台调度监控系统，并在此实践基础上展开讨论。首先，分析本文的研究背景，全面介绍与本文的相关技术基础理论与基础知识；其次，对整个通信调度监控系统进行需求分析、概要设计和详细设计；然后，利用相关技术和工具对系统进行开发；最后实现通信调度监控系统，并进行相关的测试。本文后续章节的组织结构如下：

第二章介绍了常见的几种分布式通信中间件，比如 CORBA、ACE 等，以及一些数据库方面的关键技术和理论知识。

第三章对通信调度监控系统进行需求分析，在此基础上，利用中间件和数据库等技术对整个系统进行概要设计和详细设计。

第四章完善整个轧机二级通信子系统，实现原型系统的增量开发，使用相关工具实现轧机二级通信平台调度监控系统的开发。

第五章对实现的通信调度监控系统进行本地测试与现场测试，验证本文提出方案的正确性和可行性。

第六章对整个工作过程进行总结，并依此对未来工作进行展望。

第2章 基础知识和关键技术介绍

中间件是一种独立的系统软件或服务程序，它不是某一种软件，而是一类软件的集合，分布式应用软件借助这类软件可以实现在不同技术间共享资源。本文介绍了中间件以及面向对象中间件的相关概念和特点，并重点介绍了在实际应用中几种常见的通信中间件，同时也讨论了 OO4O、DW、OLAP 等与本课题相关的技术知识，为最终完成课题提供理论基础。

2.1 中间件及面向对象中间件

中间件是基础软件的一大类，属于可复用软件范畴，伴随着计算机网络而逐步发展起来的。中间件位于操作系统、网络、数据库之上，应用软件之下，如图 2.1 所示，它可以屏蔽不同厂家的硬件平台、网络产品、网络协议的异构环境，使应用程序能够平滑地运行在不同的平台之上。对于中间件的定义，目前比较普遍被接受的表述是：中间件是一种独立的系统软件或服务程序，分布式应用软件借助这种中间件可以在不同的技术以及平台之间共享资源，中间件位于客户机服务器的操作系统之上，管理计算机资源和网络通信[11]。

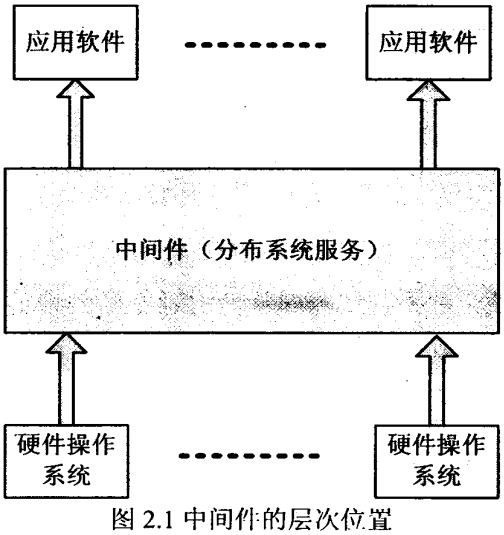

图 2.1 中间件的层次位置

Fig. 2.1 The Level of Middleware

中间件具有如下特点：

（1）满足大量应用的需要

（2）运行于多种硬件平台

（3）支持分布计算，提供跨网络、硬件和 OS 平台的透明性应用或服务的交互

（4）支持标准的协议

（5）支持标准的接口

支持标准的协议为应用程序的互操作带来便利，而支持标准的接口则大大提高了应用程序的可移植性，所以中间件逐步成为许多标准化工作的重要部分。中间件为应用程序提供了一个相对稳定的应用环境，它屏蔽了底层的差异，即使计算机硬件、系统软件或者网络服务更新换代，只要保持中间件对外接口不变，同时对中间件进行相应的更新操作，上层的应用软件几乎不需要做任何的修改即可正常使用。这大大提高了应用软件的独立性，从而降低了应用软件的开发和维护成本[12]。

近些年，计算机软件技术不断发展，中间件技术也日渐成熟，出现了很多种类的中间件产品，到目前为止，中间件的大体可分为以下六类：

（1）终端仿真/屏幕转换中间件，主要实现客户机图形用户接口与已有服务器的字符接口的匹配，实现客户机与服务器应用程序的互操作。

（2）数据访问中间件，这类中间件大部分基于 SQL 语句，采用同步通信方式，实现应用程序与数据源之间的互操作。

（3）远程过程调用中间件，这类中间件屏蔽底层网络和网络上数据传输所用的协议，使得远程调用独立。

（4）消息中间件，通过此类中间件构建的分布式应用，可将应用扩展到不同的操作系统和不同的网络环境，消息中间件可以支持同步方式和异步方式，具有较强的容错性，很好的适用于面向对象的编程方式。

（5）交易中间件，专门针对联机交易处理系统设计的中间件，例如订票系统、银行业务系统等，它是一组程序模块，其作用是减少开发联机交易处理系统所需的编程量。

（6）面向对象中间件，面向对象的中间件是对象技术和分布式计算发展的产物，它提供一个标准的构件框架，使不同厂家的软件通过不同的地址空间、网络和操作系统交互访问，这个构件框架的具体实现、位置以及所依附的操作系统对客户来说都是透明的。

本课题主要采用的中间件技术主要是面向对象中间件，下文对面向对象中间件进行了进一步的讨论。

面向对象中间件的基本思想是在对象与对象之间提供统一的接口，使对象间的操作独立于对象的位置、编程语言以及对象所依附的操作系统[13]。它可以在异构的分布式计算环境中透明地传递对象请求，无论这些对象是处于本机或者处于远程计算机。面向对象中间件的目标就是为软件用户以及开发人员提供一种透明的、即插即用的互操作模式。

类似于 OSI 参考模型中对网络通信工作的分层结构，面向对象中间件也分为四个层次，如图 2.2 所示。

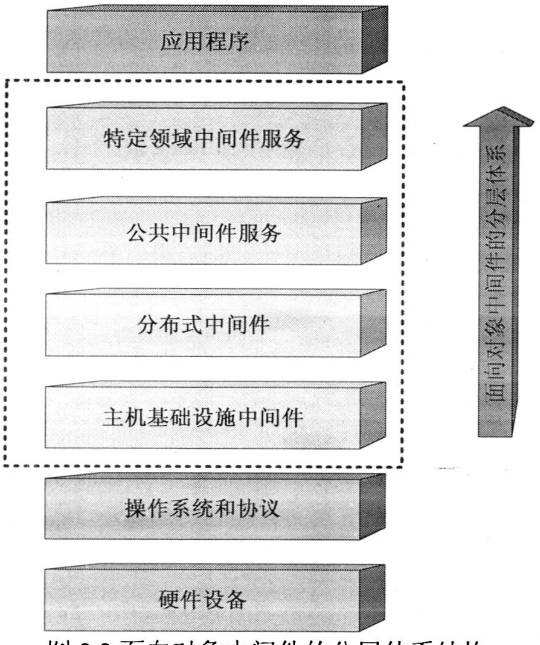

图 2.2 面向对象中间件的分层体系结构

Fig. 2.2 The Hierarchical Architecture of Object-Oriented Middleware

（1）主机基础设施中间件（Host infrastructure middleware）：通过封装操作系统的并发（concurrency）机制和进程间通信（IPC）机制，获得面向对象的网络编程能力，这样以来通过原始的操作系统 API 的调用所带来的易错性、不可移植性等问题都可得以避免。目前，主机基础设施中间件已广泛应用于 ACE 以及 Java Package 中。

（2）分布式中间件（Distribution middleware）在主机基础设施中间件基础上，对其进行了扩展，增加了包括连接管理、内存管理、整编（marshaling）、解编（demarshaling）、同步和多线程等网络编程行为，使得一些网络编程任务自动化。请求对象只需向目标对象提出请求操作，不再关心目标对象所处的位置、所用的编程语言、所依附的操作系统以及所使用的硬件环境，程序员对于这样的分布式应用程序的开发如同开发一个独立的应用程序一样简单方便。ORB（Object Request Broker）是分布式中间件的核心，常见的分布式中间件有：Java RMI，CORBA 等。

（3）公共中间件服务（Common middleware service）对分布式中间件又进行了扩展，定义了例如日志记录、安全可恢复事物、事件通知等高层次服务。同时，公共中间件服务可以对整个分布式系统中的各种资源进行分配调度，弥补了分布式中间件主要支持“面向对象”分布式编程模型的终端管理的不足。常见的公共中间件服务有：Sun 的 J2EE、Microsoft 的.NET 等。

（4）特定领域中间件服务（Domain-specific middleware service）为一些特定领域，如电子商务、电信、航空电子等提供特定的服务。

面向对象中间件是网络应用程序开发的重要工具，与传统开发网络应用相比，它主要有三方面的改进：

（1）以战略为中心。不必过于关心底层操作系统及网络，使得开发者把主要的精力放在如何满足客户应用程序要求的战略高度上来。

（2）有效的可复用性。利用可复用的中间件框架，很多网络应用程序的开发工作得以简化，降低开发成本。

（3）开放的标准。有了开放的标准，使得开发出来的产品具有较好的可互操作性和较强的可移植性特点。有助于将开发者的注意力引导到高级别的软件设计问题上，而摆脱各个不同厂商因开发标准不同带来的不便。

2.2 分布式中间件 CORBA

CORBA（Common Object Request Broker Architecture，公共对象请求代理体系结构）是由 OMG（Object Management Group，对象组织管理）提出的一个的分布式对象计算规范，是在当今快速发展的软件与硬件资源的情况下发展出的一种新技术，其目标是解决分布式计算环境中软件和硬件系统互连问题[14]。CORBA 可以让分布的应用程序完成通信，无论这种应用程序是什么厂商生产的，只要符合 CORBA 标准就可以相互通信。CORBA 1.1 于 1991 年由 OMG 提出，同时还提出了 IDL（Interface Definition Language，接口定义语言）和 API（Application Program Interface，应用程序接口），从而能够让客户/服务器对象在特定的 ORB（Object Request Broker，对象请求代理）实现中进行通信。1994 年 OMG 又提出并采纳了 CORBA 2.0 标准，它定义了如何跨越不同的 ORB 提供者进行通信，真正实现了不同生产厂商间的互操作性。CORBA3.0 是最近发布的，引入了组件模型等许多新技术为推广分布式应用打下了基础[15]。

COBRA 主要由公共对象服务、公共设施、应用对象和 ORB 四部分组成[16]。

（1）公共对象服务（Common Object Services）为使用和实现对象而提供了基本的、系统级的服务集合，扩充完善 ORB 的功能。主要的服务有：命名服务（Naming Service）、关系服务（Relationship Service）、事件服务（Event Service）、事务服务（Transaction Service）等，这些服务几乎包括了分布系统和面向对象系统的各个方面。

（2）公共设施（Common Facilities）提供可直接为许多不同业务对象使用的服务，例如：数据库操作工具、文本打印工具、时间工具、文件管理工具等。它是一个复杂的、可重用的构件块，通过规定业务对象有效协作所需的协定规则实现跨领域构造应用程

序，用户接口、信息管理、系统管理这些都是典型的公共设施。

（3）应用对象（Application Object）是由开发商借助 ORB 公共对象服务及公共设施开发的产品，用于它们的接口，不在 CORBA 体系结构中实行标准化，不属于 OMG 标准的内容。

（4）OBR 是 CORBA 规范的核心内容，在 CORBA 中，无论是本地通信还是远程通信，都要通过 ORB 进行，对象在分布的环境中透明的收发请求和响应，对于与之通信的其他对象来说都是等价的。因此，ORB 可屏蔽不同对象的通信机制、程序语言、网络环境以及对象的物理位置等因素，保证处于异构分布式环境中的不同对象系统间的无缝连接。

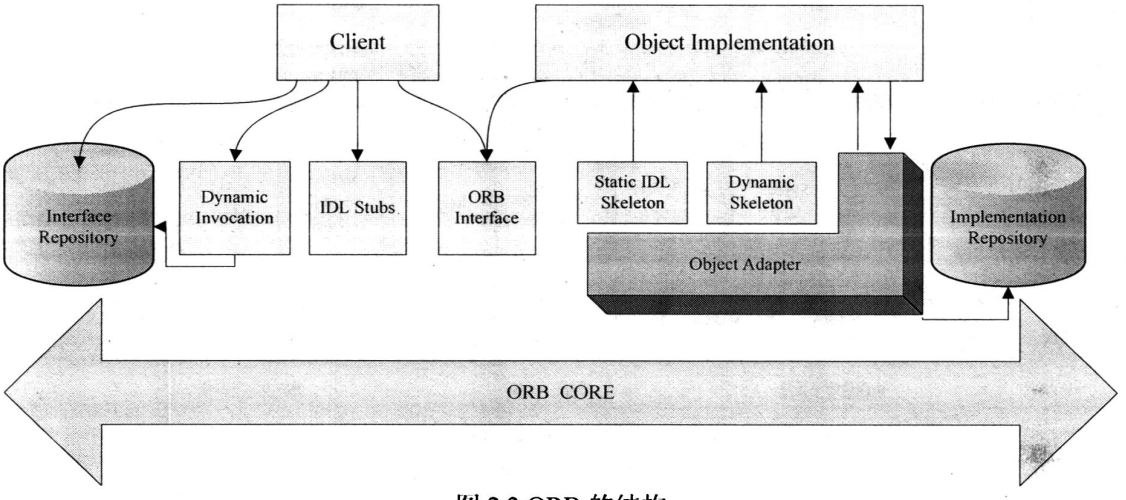

图 2.3 ORB 的结构
Fig. 2.3 The Architecture of ORB

OBR 的基本结构如图 2.3 所示，根据图中箭头指明的调用关系，对象通过 IDL 定义可以将对象的定义和对象的实现独立开来，最终使得对象的调用和实现可以使用不同的编程语言。IDL 编译器将对象的 IDL 文件编译成客户端的存根和服务器端的框架，客户端对象可以根据存根静态调用或者根据 IDL 描述信息动态调用可使用的 ORB 的服务；而服务器端对象在执行客户端对象请求时，通过对象适配器获取 ORB 服务。ORB 的这种结构使得它具备了高级语言绑定、位置透明、内置的安全检查和事务处理等优点。

2.3 自适应中间件 ACE

ACE（Adaptive Communication Environment，自适配通信环境）[17]是一个强大的 C++工具包，它实现了许多用于并发通信软件的核心模式，能帮助程序员更轻松、更迅速地开发可移植、高性能的应用程序，尤其是在网络化和多线程化的应用中，利用它可使程序具有更高的灵活性及更少的错误数量。同时，ACE 的各种核心框架和 wrapper

facade（包装外观）提供了一组集成的可复用面向对象类，能够简化且自动化一些事物的处理，用少得多的代码便可以完成所要做的工作，这样以来，C++通信软件开发将更加简单，并具有较强的灵活性，高效性和可移植性。

ACE 可跨多种平台提供服务[18]，其中包括：事件多路分离和事件处理器分派、信号处理、服务初始化、进程间通信、共享内存管理、消息路由、分布式服务动态（重）配置、并发执行和同步等。ACE 不是一个类库，它是一个强大的、面向对象的应用工具包。ACE 工具包的设计采用了分层结构，如图 2.4 给出了 ACE 的层次结构和主要组件，主要分为以下几个层次：OS 适配层、C++包装层、框架层和网络化服务层。

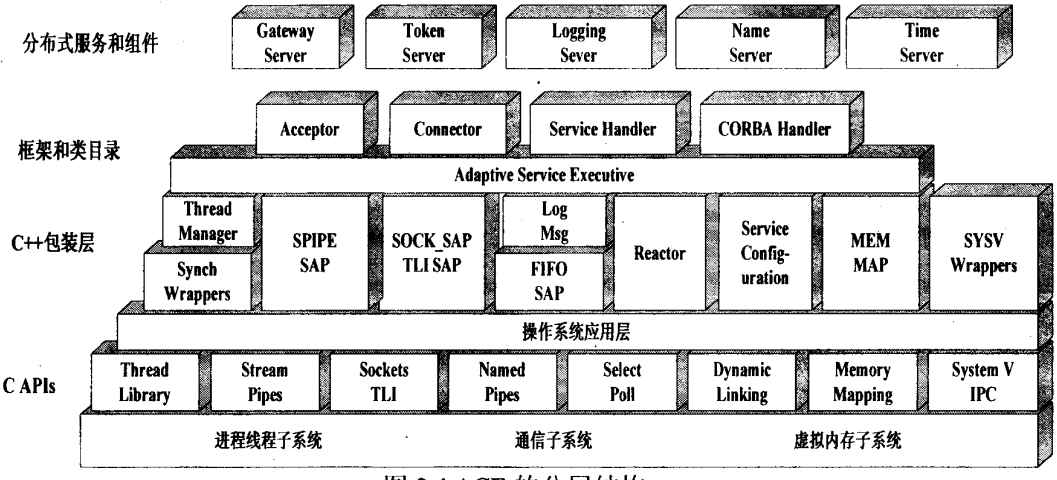

图 2.4 ACE 的分层结构

Fig. 2.4 The Hierarchical Structure of ACE

（1）OS 适配层是位于本地 OS API 和 ACE 之间的"瘦"代码层，为最常用的系统级操作提供包装函数，当原生平台没有提供所需函数时，该函数会被 OS 适配层模拟出来，如果有本地函数可用，调用通常就会内联，以使得性能最大化，这也降低了对平台的依赖性。

（2）C++包装层包括一些 C++包装类[19]，它们可用于构建高度可移植的和类型安全的 C++应用，应用可以有选择地对它们进行集成、聚合以及实例化。这些 C++包装类主要用于并发和同步、进程间通信和内存管理等方面，还有一些类用于灵活地管理进程间共享内存。

（3）ACE 框架是一组集成的组件，是 ACE 层次结构中的最高层，组件间相互协作，为相关的应用族提供可复用的架构。这就使得设计者利用这些框架组件能够站在一个较高层面上对整个系统进行思考和构建，而不必考虑底层具体的处理过程。其中的大型组件主要有：事件处理、连接或服务初始化组件、流组件和服务配置组件[20]。

（4）网络化服务层提供了一些完整的、可复用的服务，其中包括最常见的分布式

日志服务等。

因此，ACE 的使用会为程序员的开发工作带来诸多好处[21]，例如：增强程序的可移植性、较好的软件质量、更高的编程效率、更容易向标准的高级中间件转换等。

2.4 The ACE ORB

TAO（The ACE ORB）[22,23]是美国圣路易斯华盛顿大学 Douglas C.Schmidt 教授领导的 DOC Distributed Object Computing 团队研制出的一种在自适应通信环境 ACE 基础上的高性能实时 CORBA 中间件框架，它包含大量的 C++组件，实现高性能实时通信[22]。TAO 将 CORBA 中间件、操作系统的 I/O 子系统、通信协议和网络接口相结合，提供了端到端的 QOS（Quality Of Service，服务质量）保证。

图 2.5 是 TAO 的组织结构图，它的核心是 ORB，是 C/S 联系的中介，把接口和对象实现分离开来，自动完成通用的网络编程任务，简化了异构分布式环境下系统的开发。ORB 在面向对象系统中提供了跨硬件平台、跨操作系统、跨编程语言和跨网络网络协议等远程调用功能，对那些需要分布式调用、位置透明、面向对象可重用的应用来说，ORB 是很好的选择[24]，另外它还支持异步调用、并行处理等。

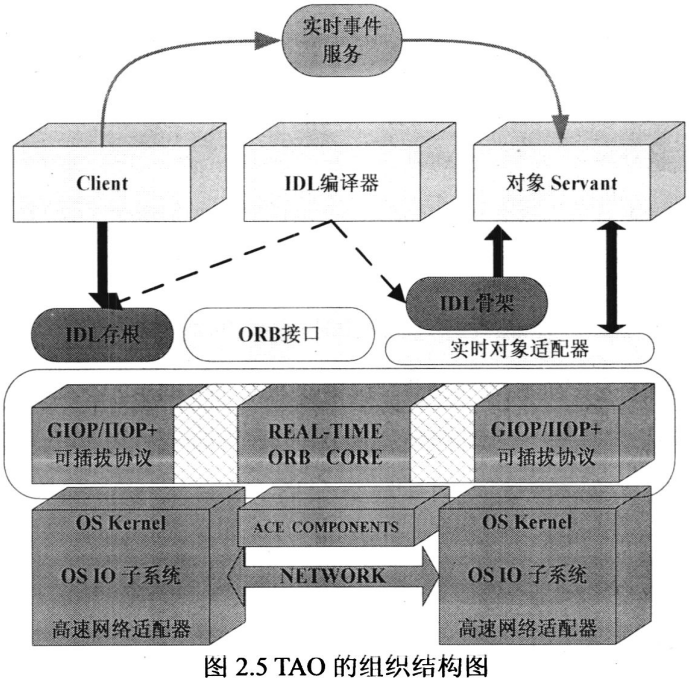

图 2.5 TAO 的组织结构图

Fig 2.5 The Architecture of TAO

以下是 TAO 结构中，几个关键组件的介绍：

（1）IDL Compiler

TAO 的 IDL Compiler 是建立在 SunSoft IDL Compiler 的增强版本上。实现了最新的 CORBA2.X IDL-to-C++映射,包括 POA。该 IDL Compiler 同时还支持 Object-by-Value 以

及 CORBA Messaging 标准。另外，TAO IDL Compiler 还可以生成实现异步方法调用的 Client 和 Server 需要的代码。

（2）内部 ORB 协议引擎(Inter-ORB Protocal Engine)

TAO 包含一个高度优化的通讯协议引擎，它实现了 CORBA 2.X 的 GIOP/IIOP 1.0,1.1 和 1.2。因此，TAO 可以无障碍的同其它支持 IIOP 的 ORB 互操作。TAO 的协议引擎支持静态和动态的 CORBA 编程，比如可以分别支持 SII/SSI 和 DII/DSI，TAO 还支持 Dynamic Anys。

（3）ORB 核心

TAO 核心为高性能和实时应用提供高效的、可扩展以及 predictable 的单向、双向以及可靠的单向同步或异步传输架构。它提供如面的同步模型： reactive、线程每连接(thread-per-connection)、 线程池(thread pool)以及 reactor-per-thread-priority(该模型是为实时系统专门优化设计的)。在以上所有的同步模型中，TAO ORB 核心支持嵌套的向上调用。

（4）可移植适配器(POA，Portable Object Adapter)

TAO 的 POA 设计采用了高优化的请求分发策略，主要是为了解决服务器端的对象实现的可移植问题。POA 对象由 POA 管理器负责创建，POA 管理器还负责将调用传给合适的 POA，交由其处理，POA 可以选择不同的策略，这些策略可以看做是一个 POA 的配置属性。

（5）实现库（Implementation Repository），用它可以实现自动定位客户端请求的目的服务器。

（6）接口库(Interface Repository), 提供 IDL 接口的运行时信息。

系统运行时，客户调用服务器对象的方法，ORB 截获此调用后，寻找能够实现此方法的对象，把参数传给对象，调用对应的方法并返回结果。客户不知道对象是否在同一机器上，也不必知道其编程语言和所在网络类型，总之就是不必知道任何该对象接口以外的其他信息。因此，ORB 提供了不同机器上组件间的互操作，实现分布式、异构组件间的无缝连接。

2.5 Oracle 数据库访问技术 OO4O

Oracle Object for OLE (简称 OO4O)是 Oracle 8i 以上版本提供的一个强大的工具，它可以无缝连接并优化对 Oracle 数据库的访问[25]。OO4O 支持 Microsoft 公司的 COM Automation and ActiveX 技术，且支持多种编程语言，包括：Visual Basic、Visual C++、Visual Basic For Applieations(VBA)和 IIS Active Server Pages(VB Script and Java Script)。

OO4O 也可用于从 web 应用到 n 层 C/S 应用程序的多种环境。OO4O 主要由三部分构成：进程内自动化服务(In-process Automation Server)、Oraele 数据控件(Oracle Data Control)和 C++类库(C++ Class Library)[26]。

（1）进程内自动化服务(In-proeess Automation Server)

进程内自动化服务提供了一套 COM 自动接口对象，这个接口对象可以方便地完成在应用程序中的数据库连接，以及对数据库查询、管理等工作。

（2）Oracle 数据控件（ODC）

Oracle 数据控件是一种简化查询结果和常用可视化控件之间数据交换的 OCX 或者 ActiveX 控件。

（3）OO4O 的 C++类库

OO4O 把所有的自动对象都封装到自己的 C++类库中，简化了自动对象的获取，使得程序员用 C++语言可以快速的访问 Oracle 数据库。在 Visual C++中使用 OO4O 访问 Oracle 数据库就是通过 OO4O 的 C++类库实现的，它封装了 Oracle 数据库的主要 API 函数，常用的类主要有以下几个：

（a）ODatabase 类，是用来管理数据库连接的，可以通过 ExecuteSQL 直接执行 SQL 语句。

（b）ODynaset 类，是用于创建并管理数据库的记录集。

（c）OField 类，是用来对数据库表中的字段进行管理，使用 GetValue 方法可以获得字段的值，使用 SetValue 可以设定某个字段的值。

（d）OParameter 类与 OParameterCollection 类，是对数据库参数集进行管理的两个类。

（e）OException 类是对数据库操作的异常情况进行处理的类，可以获得异常操作的反馈信息。

2.6 DW+OLAP

数据仓库技术（Data Warehousing，简称 DW）、联机分析处理技术（On-Line Analytical Processing，简称 OLAP）是一个商业智能系统（Business Intelligence System，简称 BIS）的最基本也是最重要的部分。随着数据库技术的应用与发展，人们更注重对数据库中长期存储的大量数据的分析加工、形成一个综合的、全面的、细致的数据信息库，以更好的支持决策分析，所以逐步形成 DW 与 OLAP 的结合与发展。

2.6.1 数据仓库 DW

数据仓库[27]是面向主题的、集成的、稳定的、反映历史变化的数据集合，用以支持管理决策。其中面向主题、集成性、稳定性、历史性也是数据仓库的四大基本特征[28]。

（1）面向主题是数据仓库的第一大基本特征，主题是一个抽象的概念，在较高层次上对数据进行抽象处理，数据从外部数据源进入数据仓库后，在某个主题的引导下，经过抽取、转换、加载等必要的变换后以适用的方式存放起来。

（2）集成性，数据仓库中的数据可能来自于不同的数据源，这些数据可能使用的是不同的的数据结构、不同的编码方式或者不同的命名规则等，而数据仓库则将导入的数据源按照一定的方法处理成一系列单一的、全局可接受的格式存储，使得所有存在于数据仓库中的数据具有一致性。

（3）稳定性，操作型数据库中的数据根据应用程序的需要经常会有更新操作，甚至一个很小量级的时间段内数据都会发生变化，而数据仓库主要功能是为分析决策提供帮助信息，一般情况只进行数据查询操作，通常只需要定期的加载即可。

（4）历史性，数据仓库中的数据记录了一段时间内的数据信息，通过这些长期存在的数据信息，可以分析决策未来的发展趋势，这是一段历史的过程。

根据数据仓库的这些特点，不难总结数据仓库与操作型数据库中存放的数据的一些不同之处，如表 2.1 所示。

表 2.1 数据库与数据仓库中数据的差别

Table 2.1 The difference between the data in database and data warehouse

类别	原始数据\操作型数据	推导数据\DW 数据
时间性	存在时间短，经常变化	长期存在，相对稳定
集成度	细节数据	综合的聚集数据
可更新性	实时更新	很少更新
驱动方式	事件驱动	分析驱动
对性能的要求	对性能要求高	对性能要求宽松
存取概率	存取概率大	存取概率低、中等
冗余性	非冗余	冗余是存在的事实
规模	几个 GB	可以超过上百个 GB

2.6.2 联机分析处理 OLAP

联机分析处理[29]是专门用于支持复杂分析操作的、快速的、灵活的进行大数据量复杂查询处理的软件技术。它为分析人员提供对数据的多角度观察和一致性交互操作，使得分析人员可以获得对数据信息的深入理解。

OLAP 主要包括以下几个特性：

（1）多维性，OLAP 支持用户进行多维分析，其中包括对层次维和多重层次维的建模，对多维数据集中的数据进行切片、聚合、旋转、钻取等操作，剖析数据的背后蕴含的规律，多维性是 OLAP 的灵魂。

（2）在线性，是指 OLAP 可以满足用户快速查询分析的要求，在短时间内按用户的要求对数据进行转换处理，并给予反馈。专门的数据存储格式、大量的事先运算、特别的硬件设计为在线性的实现提供了技术保证，在线性提高了数据分析的质量。

（3）可分析性，是指系统必须处理与应用有关的任何逻辑分析和统计分析，用户可以在 OLAP 平台上进行数据分析，也可以连接到其他外部分析工具上分析，如时间序列分析工具、成本分配工具等。

（4）信息性，是指 OLAP 系统可以及时获得大量的信息，这里需要考虑数据的可复制性、可用磁盘空间，OLAP 与数据仓库的结合度等。

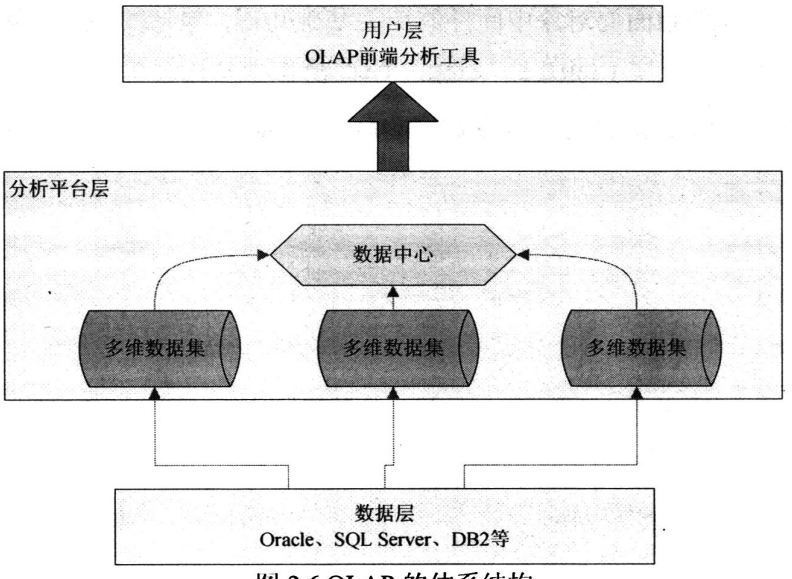

图 2.6 OLAP 的体系结构

Fig. 2.6 The Agriculture of OLAP

OLAP 主要包括 MOLAP、ROLAP、HOLAP 三个类[30]。

（1）MOLAP（Muttidimensional OnLule Analytical Processing），数据以多维方式存储，通过维度的定位可以访问每一个数据单元。MOLAP 具有响应迅速、支持高性能决策等优点，但由于实际数据分布稀疏，会导致数据急剧膨胀，增加系统复杂度。

（2）ROLAP（Relational Online Analytical Processing）

ROLAP 可以将用户的多维查询请求转化为 SQL 查询，又将查询结果以多维的方式呈现给用户,这样使得现有的关系数据库技术得以沿用，提高 ROLAP 的速度。然而相对于 MOLAP 来说，其响应速度还是相差甚远，而且 SQL 也无法完成所有的计算工程。

ROLAP 中的数据常以星型模式或者雪花型模式存储。

（3）HOLAP(Hybrid Online Analytical Processing)

HOLAP 是 MOLAP 与 ROLAP 的结合形式，兼具 MOLAP 的查询效率高和 ROLAP 存储效率高的优点。

如图 2.6 所示，OLAP 系统主要分为三个层次：数据层、分析平台层和用户层。数据层提供了 OLAP 分析的数据基础，它们可以来自不同的数据源，例如 Oracle、DB2、SQL Server 等。数据层将数据传递给分析平台层，分析平台层根据获得的数据和生产需求设计 OLAP 数据库，通过数据的抽取、转换，建立数据中心，再根据不同视角建立数据集市，为用户层的分析做好准备。用户层则利用前端显示工具，对分析平台层数据中心中的数据进行 OLAP 操作，获得有价值的信息。

2.7 本章小结

本章介绍了中间件和面向对象中间件的相关基础知识，包括其定义、体系结构和实际应用领域等，重点讨论了 CORBA、ACE、TAO 的框架结构和各自的特点。同时介绍了数据库访问技术 OO4O 的优势，以及数据仓库、OLAP 的基本概念和特点。

第3章 通信调度监控系统的分析与设计

本章对中厚板轧机二级通信平台调度监控系统进行了需求分析，然后根据需求分析的结果对系统进行概要设计。本章主要将整个系统分为轧机二级通信子系统、轧机二级监控子系统和轧机二级自学习子系统三个部分进行详细设计，并在此基础上构建完整的中厚板轧机二级通信平台监控调度系统。

3.1 中厚板轧机二级通信平台调度监控系统的需求分析

轧机二级是整个生产线的核心，不仅需要对一级基础设备测量的数据进行收集、计算，同时还要满足三级提出的任务要求，是实现数据共享、自动化生产的重要环节。在实际的中厚板生产线上，轧机二级所处的位置如图 3.1 所示。

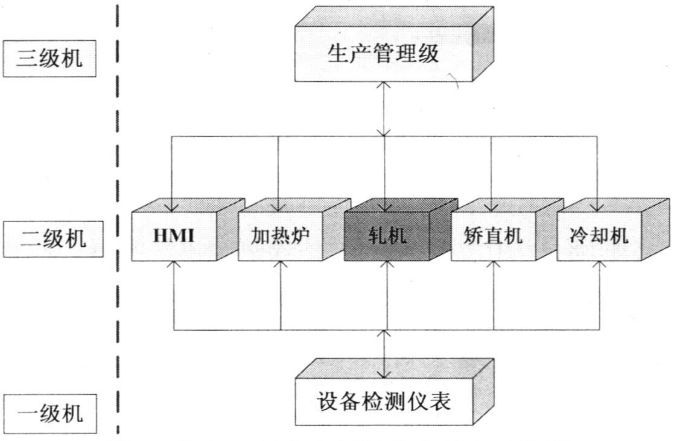

图 3.1 轧机二级在中厚板生产线中的位置

Fig. 3.1 The place of mills level 2 in a medium plate production line

因此，要真正的实现中厚板生产线的自动化生产，轧机二级系统需要具备如下几个功能：数据通信、数据处理、轧制跟踪、模型自学习等。

（1）数据通信

数据通信是实现轧机二级与其它设备系统连接的基础，轧机二级与其他外部设备或者服务器之间进行通信交换的主要方式是工业以太网的 TCP/IP 传输、基于内存映射文件的共享内存和消息队列等。首先，轧机二级系统需要与一级基础自动化设备进行通信，轧机一级计算机收到来自轧机二级计算机的数据后，对中厚板生产线上的各种轧制设备（如加热炉、冷却机等）的值进行设定，同时轧机一级经过实际运行后，再把各种轧制设备的实际测量参数返回给轧机二级；其次，轧机二级系统还要与三级生产管理系统进行通信，轧机三级收到轧机二级发来的与产品质量相关的数据，在对这些产品数据分析后，基于提高产品质量的角度考虑，对轧制参数做出优化调整，并返回给轧机二级计算

机。轧机二级成为基础自动化级和生产管理级的桥梁和纽带，因此，轧机二级进程调度和参数传递的正确性是实现轧机二级数据通信的必要条件。

（2）数据处理

通过数据通信，轧机二级系统主要会收到两大类数据。一是由一级基础自动化级或三级生产管理级发送的参数数据，包括钢板原始数据、工艺数据、一级实测数据等，这些数据类型广、数量多，必须进行必要的处理才能继续使用，另外对于一些重要数据还需要存入数据库，为模型计算和系统监控提供数据基础。二是由其它二级系统或者设备发送的响应数据，这些数据主要是用来触发轧机二级的某些进程开启特定的功能模块，轧机二级收到这些数据，根据预先定义的规则，执行特定的功能。

（3）轧件跟踪

在中厚板的生产线上，为了节约成本，提高轧机的利用效率，可能会有多块轧件在同一时刻处于轧机的待轧区域，尤其在进行多坯交叉轧制的时候，在待轧区域中，每块轧件的加工状态可能各不相同，有的处于轧制状态，有的处于待温状态等。要想正确的对每块轧件进行轧制，必须区分每块轧件的加工状态、所处位置，因此，轧机二级需要对每一块轧件进行跟踪处理，跟踪功能也是轧机二级系统的重要功能之一。

轧机二级根据来自基础自动化级的相关设备如热金属检测器、检测仪表（测温仪、力传感器）等发出的信号和当前状态下辊道运转、轧机运行等发出的控制信号来实现轧件的跟踪。通过跟踪功能，可以正确记录当前轧件的位置和轧制状态，并根据当前的实际情况做出相应的处理。

（4）模型自学习

在实际的中厚板生产线上，轧机二级系统的模型设定值往往存在不可避免的误差，这些误差产生的原因可能来自多方面因素，比如测量误差、机械误差等，这些误差复杂多变，很难单一的控制，往往需要再次对实际生产条件进行计算才能给出正确的模型设定值，然而这恰巧违背了自动化生产的目标。

模型自学习是根据一个轧制过程获得的实测数据甚至每一个已完成道次的实测数据对模型进行修正，这些数据常常来自于数据库系统中，自学习模型可以消除同类钢板的共同性误差，实现中厚板的自动化生产。

根据上文提到的中厚板轧机二级系统功能需求，构建一个具备多进程调度能力、数据存储、数据分析、进程监控等功能的轧机二级通信调度监控系统具有重要意义。本文提出的中厚板轧机二级通信平台调度监控系统，从中厚板生产线实际情况出发，致力于东北大学轧制技术及连轧自动化国家重点实验室开放课题《中厚板轧机二级控制系统开

发》，大体提出如下几方面的系统需求：

（1）高效的多进程通信

在实际生产过程中，轧机二级处于生产线的核心位置，需要与轧机一级、轧机三级甚至其他二级服务器包括加热炉服务器、测厚仪服务器等进行通信，以实现自动化控制生产，这就使得系统中存在多进程的数据通信，多进程通信往往带来数据不一致、数据延迟错误等问题，如何保证多进程通信中，各个进程收发数据的正确性与实时性，是整个通信调度监控系统的重要功能需求。

（2）系统的稳定性

必须保证整个系统的稳定性，保证每个进程正确的运行，防止进程僵死而导致生产过程中的停机、宕机等意外情况的发生，造成不必要的损失。

（3）数据永久性存储

系统运行过程中的重要数据需要做的永久性存储，一方面要为计算模型提供正确的数据基础，另一方面也要为轧机二级自学习子系统的数据分析工作提供数据基础。另外，数据存储的正确性和实时性也是一个重要的问题。

（4）良好的数据分析能力

在中厚板的生产过程中会产生大量的数据，这些数据可能存在潜在的联系，不能只对这些数据进行简单的存储，还需要利用海量数据对其进行分析，发现其潜在的规则，为自学习子系统提供有用的数据信息。

（5）良好的监控能力

要想了解系统是否正确的运行，必须对系统进行实时的监控，这就需要一个监控子系统来解决进程的调度和系统监视等问题。监视画面要简单易操作，能够让工作人员一目了然的发现问题并解决问题。

（6）可扩展性

本文提出的中厚板轧机二级通信平台调度监控系统，是在东北大学轧制技术及连轧自动化国家重点实验室开放课题《中厚板轧机二级控制系统开发》前期完成的原型系统的基础上进行进一步的设计与开发，系统在以后的使用中还需要根据实际生产的功能要求进行更深层次的改进，所以本系统应该具备良好的扩展性，为以后的开发工作带来方便。

（7）可维护性

实际生产过程中会出现许多预想不到的问题，有些严重的问题可能会使得系统要根据需要进行一定的变动，所以系统需要具备良好的可维护性，以便后期的工作人员方便

的进行部署。

上文对中厚板轧机通信平台调度监控系统进行了需求分析，研究设计本系统主要有以下几个难点：首先是多进程通信的调度问题，如何保证各个进程的互斥与同步，保证各个进程间数据交换的正确性与实时性是整个系统正确运行的重中之重。其次是对数据的管理问题，怎样实现数据的永久性存储和对海量历史数据进行有效的分析，是保证系统可行性的关键。最后是系统监控问题，怎样设计一个能够及时反映系统变化情况的监控子系统是维护系统正确运行的保障。

本文所讨论的是中厚板轧机二级通信平台调度监控系统的设计与实现，是在东北大学轧制技术及连轧自动化国家重点实验室开放课题《中厚板轧机二级控制系统开发》前期完成的原型系统的基础上进行进一步的研究与开发，完成轧机二级通信子系统与监控子系统和自学习子系统的结合，实现最终的中厚板轧机二级通信平台调度监控系统。

3.2 中厚板轧机二级通信平台调度监控系统的概要设计

通过对中厚板轧机二级通信平台调度监控系统的需求分析，可以看出轧机二级通信调度监控系统在多进程通信、对数据实时性、历史数据分析能力、进程的监控、分布式服务、日志处理等方面都有很高的要求。通过对 DW、OLAP 技术调查和对中间件技术 CORBA、ACE、TAO 的进一步技术研究，结合中厚板轧机二级系统的实际需求、项目前期开发人员的技术经验，将整个系统分为轧机二级通信子系统、轧机二级监控子系统和轧机二级自学习子系统三个部分进行开发，最终完成中厚板轧机二级通信平台调度监控系统的开发工作。

3.2.1 系统框架的设计

通过本文前面的讨论分析，大体了解了本系统开发的难点在于多进程通信的正确调度问题，包括进程的互斥、同步以及死锁问题。由于 C++本身不提供解决这类问题的库函数，所以要完成多进程通信的任务大部分都要靠系统的 API 函数支持，这样不仅增加了开发难度，更会影响系统的可移植性，使其很难再不同的操作系统下正常的运行。然而，ACE 为多进程通信提供了安全性保护[31]，首先 ACE 使用了互斥体，这是 ACE 中最简单的保护原语，它提供了简单的 acquire()和 release()接口，如果成功获得互斥体，进程继续执行，否则就会阻塞，直到互斥体所有者使用 release()释放它为止；再者，ACE 使用了守卫，在许多情况下，异常状况会在本可以完好运行的代码中造成死锁，例如忘记释放互斥体就会造成死锁，ACE 提供了一个 ACE_Guard 类，它基于常见的 C++手法：把构造器和析构器用于资源获取和释放，守卫类在构造时获取一个指定的锁，在销毁时

释放这个锁，如果把守卫放在栈上，就可以总能保证锁的释放。另外 ACE 还提供了日志服务，每个程序都需要诊断信息，例如错误消息、调试输出等，传统上为了帮助跟踪执行路径，常使用许多 printf()调用或 cerr()调用，这种方法常常难以控制、操作起来相对复杂，ACE 的日志设施提供了多种完成这些任务的途径，对诊断信息的打印和定向有着强有力的控制能力，常用于诊断输出的宏有三个：ACE_DEBUG、ACE_ERROR 以及 ACE_TRACE。因此，本文基于 ACE 中间件来进一步设计开发轧机二级通信子系统，从而弥补 C++语言本身的缺陷，增强系统的稳定性和健壮性。

C++是面向对象的编程语言，CORBA 是一个跨网络、跨语言的分布式计算平台，它们具有自己独立的技术特点，适用于解决异构系统的集成问题[32]。但是 C++语言不支持远程对象调用、不支持跨操作系统的运行，需要一个支持异构网络通信的底层框架弥补这一缺陷，CORBA 恰是一个比较成熟的分布式计算平台，能够很好的支持跨网络的异构对象间的通信，同时 CORBA 作为一种语义规范正好需要 C++语言来具体实现其功能。因此，COBRA 与 C++语言的结合使用，可以互相补充，对系统设计与开发带来极大的便利。

TAO 是一种 C++ 实时 ORB，它兼容大部分 CORBA 标准，具有性能高、互作性强、健壮性好等特点，已在商业化项目中有广泛的应用；同时，TAO 也是基于 ACE 开发的实时的 ORB，所以 TAO 可以很好的与 ACE 无缝结合[33]。

经过上述分析，进一步证实了 ACE+TAO 通信框架的可行性，为构建轧机二级通信子系统提供技术支持。

通过本文前面的需求分析，为了保证计算模型和轧机二级自学习子系统的正常使用，必须为其提供可靠的数据基础，然而普通的数据库只能做到简单的数据存储，对数据分析方面难有作为。DW（数据仓库）能够保证“库尽其用”，这是传统数据库所不及的[34]。虽然数据仓库以数据库为基础而建立，但是它与数据库又存在很大的不同。首先，数据仓库不是大量数据的简单堆砌，而是将大量数据进行适当的抽取、转换、集成后形成的一个新的数据库系统；其次，数据仓库包含的是一系列概括性的、总结性的数据，为之后的数据分析工作提供良好的数据基础。

OLAP（联机分析处理），是适合以数据仓库为基础的分析处理，OLAP 中历史的、导出的、综合提炼的数据均来自数据仓库[35]，OLAP 将从数据仓库导入的数据镜像多维化的综合处理，建立不同级别的统计数据，从而满足快速统计分析和查询数据的要求。同时 OLAP 多采用便于非数据处理专业人员理解的方式（如图表、统计图等），让用户能够方便的进行切块、钻取、旋转的数据操作。

经上述分析研究，进一步证实了基于 DW+OLAP 框架结构的决策分析系统是一种比较理想的架构，将它合理的用于本文提出的中厚板轧机二级通信平台调度监控系统中的自学习子系统部分具有良好的可行性，使得系统功能更加齐全，可以对轧机二级子系统中的数据进行准确的分析，为实现轧机二级自学习子系统的开发奠定基础。

图 3.2 给出了中厚板轧机二级通信平台调度监控系统的组织结构图，从图中可以看出整个系统主要包括四个部分：计算模型、监控子系统、自学习子系统和通信子系统，其中通信子系统又可以细分为功能模型和通信接口。

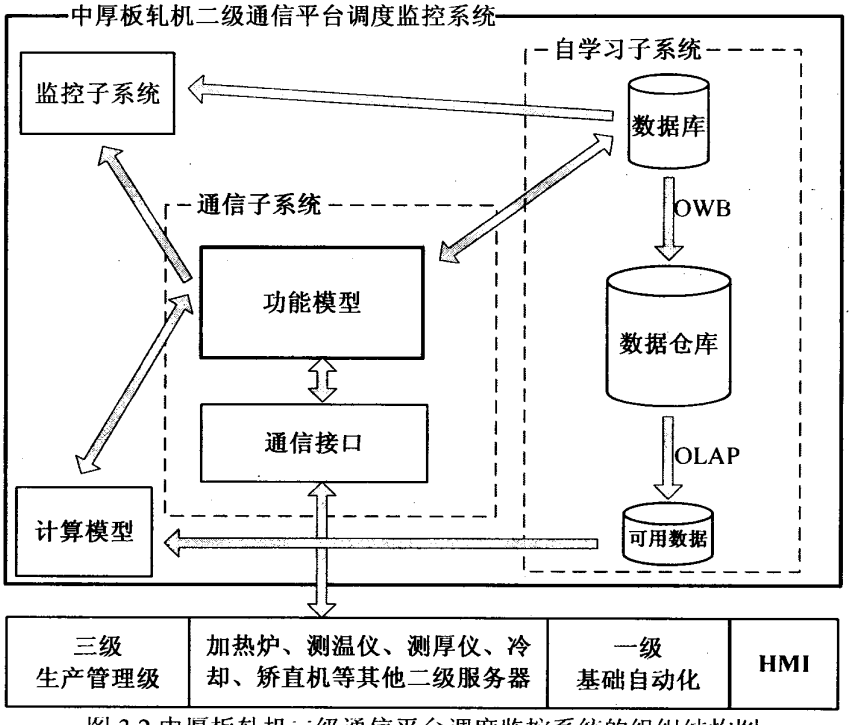

图 3.2 中厚板轧机二级通信平台调度监控系统的组织结构图

Fig. 3.2 The architecher of dispatching and monitoring system for the Medium plate mill level 2 communication platform

计算模型部分主要包括：钢板温度计算模型、道次规程计算模型等，其中道次规程计算模型又分为预计算、后计算和再计算三个部分。预计算是在轧制开始前确定完整的道次规程，例如道次数、轧机轧制力、下压力等，使其根据不同的板坯和所需的目标钢板达到预定值。后计算是在轧制过程中，根据实际测量值（轧制力、扭矩、弯辊力、轧制温度等）以及当前钢板的轧制状态，可以计算出口处钢板的实际尺寸，同时通过比较实测数据与预计算数据，可以对相关工艺模型进行短期修正。对未轧道次的设定值进行重新计算，也称为再计算，再计算可以修正道次规程，提高轧制质量。

通信子系统主要包括功能模型和通信接口两个部分。功能模型主要采用多进程通信模式，负责整个通信系统的调度控制，它是整个轧机二级通信平台的核心。通信接口是

轧机二级与一级基础自动化、其他二级服务器以及三级生产管理级通信的接口进程，负责数据的实时传输，由于系统在后期的开发使用中需要逐步完善，增加新的功能模块，所以通信接口需要具有良好的扩展性。

自学习子系统主要包括数据库、数据仓库，以及 OLAP、OWB 等数据分析工具，它不仅负责对数据的永久性存储，同时还需要对数据库数据进行分析，然后将抽取、转换后的数据提供给计算模型，为计算模型进行正确的计算提供数据基础。

监控子系统主要对系统中进程的运行状态和数据库中的数据收发进行监视控制，实时的显示系统中正在运行的进程，并可以对其进行强制的开启或关闭，同时通过对系统日志的监视，能更好的掌握系统运行状态，及早的发现系统中的错误信息从而尽早的做出相应的调整。

3.2.2 进程间通信方式的讨论

在 Windows 程序中，WIN32 API 提供了许多函数进行高效的进程间通信，通过这些函数，从而控制各个不同进程实现数据交换[36]。

常见的进程间通信方式主要有：消息队列、管道、信号、共享内存等，而这些进程间通信方式又可以概括的分为两大类，即消息传递和共享内存。消息机制实现方便、应用灵活，然而在数据量很大的情况下，数据会被拆分成若干段，以消息序列的方式进行发送，因而有数据量小、携带信息少等缺点，所以消息机制常常用于无须大量、频繁数据交换的进程通信系统中。另外消息中间件是中间件技术的一个分类，利用高效可靠的消息传递机制进行与平台无关的数据交流，通过提供消息传递和消息排队模型，屏蔽操作系统、网络环境、通信协议等差异，在分布式环境下扩展进程间的通信。共享内存通信方式允许多个进程使用同一个内存段进行通信，它将共享的内存缓冲区直接附加到每个进程的虚拟地址空间中来[37]，就像数据位于每个进程的本地地址空间一样。共享内存通信的特点是无中间环节，每一个进程可以直接访问同一个物理内存页面，就如访问自己的私有空间一样。进程的痛惜可以直接读写共享内存，相对于消息通信机制减少了数据的拷贝工作，所以采用共享内存一个最大的优点就是其高效的数据访问效率。

ACE 提供了若干工具可以帮助开发人员使用共享内存[38]，ACE 的分配器 ACE_Allocator 是一种可以提供动态内存管理机制，使用它可以动态的分配共享内存，管理程序运行中内存的分配与释放。另外 ACE 还提供了一种基于 ACE_Malloc 类模板的分配器族，主要采用 C++模板和外部多态性来为内存分配机制提供灵活性。ACE_Malloc 模板有两个主要参数：锁类型和内存池类型，锁主要用于确保在有多个进程或线程同时使用分配器时的一致性，而内存池则是分配器用来获取和释放内存的地

方。ACE 提供了若干内存池，包括用于分配共享内存的池和用于分配常规堆内存的池，表 3.1 对 ACE 的内存池类型进行了介绍和描述。

表 3.1 ACE 提供的内存池类型
Table 3.1 The types of memory pools provided by ACE

内存池名称	描述
ACE_MMAP_Memory_Pool	基于内存映射文件的内存池
ACE_Lite_MMAP_Memory_Pool	轻量级 ACE_MMAP_Memory_Pool
ACE_Shared_Memory_Pool	基于 System V 共享内存的内存池
ACE_Local_Memory_Pool	基于 C++ new 操作符的内存池
ACE_Pagefile_Memory_Pool	基于 Windows 页面分配的匿名内存区池
ACE_Sbrk_Memory_Pool	基于 sbrk（2）的内存池

本文提出的中厚板轧机二级通信平台调度监控系统中，进程间通信具有复杂行，数据交换也要保证其正确性与高效性，所以综合考虑，将采用消息通信和共享内存通信两种通信方式结合，根据各个进程自身要实现的功能特点选择不同的通信方式，从而保证整个系统的通信效率。

3.2.3 数据流分析

整个轧机二级调度监控系统中的通信子系统，只存在两种数据流向，一种是数据从各个进程流入共享内存或者数据库，另一种是数据从共享内存或者数据库流入各个进程。

（1）数据从各个进程流入共享内存或者数据库，这种数据流主要有两种用途。流入共享内存中的数据可以为计算模型提供数据基础、可以为其他进程的使用提供参数，由于数据存放在共享内存中，进程可以直接进行读写操作，进而减少了进程读写时间，提高了系统的计算效率，也对系统数据的实时性处理带来很大帮助。流入数据库中的数据主要是对其进行存储备份，另外通过数据库子系统还可以对存储的数据进行分析处理，挖掘潜在的数据信息，为计算模型提供数据基础。

（2）数据从数据库或共享内存流入各个进程，这种数据流也有两种用途。从共享内存流入各个进程的数据，主要是将共享内存中其它进程流入的更新数据或者计算模型的中间结果数据返回给各个进程，为各个进程提供参数，另外共享内存中的数据还可以用于协同计算和各个进程间的同步与互斥，保证进程通信的正确性。从数据库流向各个进程的数据，主要为各个进程提供必要的参数和系数，例如钢板的固有属性（钢板材料、钢板序列号等）。

3.2.4 数据库访问方式的讨论

随着数据库级数的发展，在数据库的访问问题上也不断进步，由传统的纯 SQL 到嵌入式 SQL，再到数据库访问技术的形成。传统的数据库访问技术成熟，但是其可移植性较低，并且使用复杂，给程序员的工作带来很大的不便，而数据库访问技术的产生与发展恰好可以解决以上问题，加强了对数据访问的灵活性与方便性。目前运用数据库访问技术不仅可以访问关系数据库，还可以访问包括文本、电子表格等多种数据源。本文提出的中厚板轧机二级通信平台调度监控系统使用的是 Oracle 数据库，常见的 Oracle 数据访问技术主要有：PRO、ODBC、OLE DB、OO4O 等[39]。

（1）PRO 是 Oracle 提供的 Oracle 数据库应用程序专用的接口开发工具，使用它可以将程序中嵌入 SQL 语句，这种程序叫做 PRO*程序，例如，假如使用的是 C 语言，则 PRO*程序为 PRO*C，类似这种嵌入 SQL 语句的 PRO*程序主要包括 PRO*Ada、PRO*C、PRO*COBOL、PRO*Fortran、PRO*Pascal 和 P RO*PL/I 六种。嵌入式 SQL 可以在高级语言的程序中直接使用，而不必通过函数调用进行数据库应用程序开发。PRO 通过预编译程序将 SQL 与高级语言结合，将 SQL 语句转换为标准的 Oracle 运行时刻库（SQLLIB）函数调用，不仅利用了 SQL 的强大功能、加强其使用的灵活性，又可以充分发挥高级语言在系统开发方面的优势。

ODBC（Open Database Connectivity）即：开放数据库互连，是为应用程序访问数据库提供的通用 API 接口，只要有了相应的驱动程序就可以对数据库进行访问[40]。使用 ODBC 访问数据库调用了标准的 ODBC 函数和 SQL 语句，数据库的底层操作则由各个数据库的驱动完成，不同的 ODBC 驱动程序保证了应用程序与数据库的相互独立。然而随着数据库功能的不断增加，现有的 ODBC 函数无法满足这些新功能的实现，所以如果使用 ODBC 就无法使用数据库的一些新特性，再者，ODBC 仅能访问关系型数据库，这无非对用户产生局限，另外 ODBC 函数众多、使用复杂、访问速度慢等也是使用 ODBC 要面临的问题。

OLE DB（Object Linking and Embedding Database）是一种基于 COM（Component Object Model）规范的接口，它不仅可以访问关系型数据库，而且可以访问非关系型数据库，使用 ODBC 的开发者很容易将其转换到 OLE DB。OLE DB 属于数据库访问技术中的底层接口，通过大量使用指针直接访问内存来提高访问数据库的效率，然而在一些其他编程语言环境下（如 Java），使用 OLE DB 就会带来很多困难，而且 OLE DB 的 API 十分复杂，细节过多，较难使用。

OO4O（Oracel Objects for OLE）是 Oracle 公司提供的快速访问 Oracle 数据库的产

品。OO4O 由三部分组成：进程内的 COM 自动化服务、C++类库和 Oracle 数据控件，OO4O 虽然只能访问 Oracle 数据库，对其他的数据库不兼容，但是其一系列的优化特性可以更加方便高效的对 Oracle 数据库进行访问，OO4O 支持多种编程语言，包括：VB、C++、VBA 等，另外，OO4O 是专门为 Oracle 数据库量身定做的，所以它支持 Oracle 数据库的所有特征[41]，如支持对象关系和 LOB 数据类型，支持 PL/SQL、支持数组处理和服务器方查询等，快速访问 Oracle 数据和对 Oracle 数据库功能实现能力强是它的重要特点。

在 Visual C++环境下开发 Oracle 数据库应用程序时，以上的几种数据访问方法都可以选择，但是每种方法都有各自的优缺点如表 3.2 所示，OLE DB 具有良好的通用性但是却不能使用 Oracle 的高级特性，PRO 的性能很高但却处于底层的控制层次。由于本文提出的轧机二级通信调度监控系统对数据库访问的实时性要求较高，在结合实际开发条件和对各种数据库访问方式的进一步研究后，决定采用 Oracle10g 数据库以及 OO4O 数据库访问方式对系统进行开发，以保证数据的永久性存储、提高数据库访问的效率。

表 3.2 几种数据库访问方式的比较

Table 3.2 The comparison of several database access methods

项目	PRO	ODBC	OLE DB	OO4O
性能	很高	低	较高	较高
通用性	只适合 Oracle 数据库	适合所以关系型数据库	适合所有数据库	只适合 Oracle 数据库
控制层次	底层	高层	底层	高层
能否使用 Oracle 高级特性	能	不能	不能	能
编程复杂度	简单	较简单	复杂	中

3.3 中厚板轧机二级通信平台调度监控系统的详细设计

根据本文 3.1 节对中厚板轧机二级通信平台调度监控系统的需求分析和本文 3.2 节所提出的系统概要设计方案，再结合中厚板生产线的实际需求，本节对中厚板轧机二级通信平台调度监控系统进行了详细设计，根据不同的服务功能又将整个系统的设计划分为三个部分：轧机二级通信子系统的详细设计、轧机二级监控子系统的详细设计和轧机二级自学习子系统的详细设计。

3.3.1 轧机二级通信子系统的设计

轧机二级通信子系统中包含多个进程，大体上可分为接口进程、调度进程和计算模

型进程三大类，本节将主要针对这三类进程间的通信关系以及它们对数据库的访问情况进行进一步的详细设计。

3.3.1.1 接口进程与调度进程通信的设计

接口进程主要负责完成轧机二级与外部服务器的通信，按照各自实现的功能不同，主要有以下 7 个接口进程：l3l、ful、hmil、il1、ol1、pgl、accl。调度进程是整个系统的关键部分，主要负责数据的分析处理，根据来自接口进程的数据判断钢板的轧制状态并加以修正和维护，同时对计算机模型进程进行调度。调度进程主要有以下 6 个：dispatch、papSnd、papRcv、meas、spt、dbwriter。随着系统的不断升级，还可以根据实际需要进行添加。

接口进程与调度进程有着密切的通信关系，主要有以下几种形式：

（1）如图 3.3 所示，接口进程 l3l 负责与三级服务器通信，l3l 接收三级服务器发送的板坯原始数据，通过 TAO 调用相关函数，触发调度进程 papRcv。接口进程 ful 负责与加热炉二级服务器进行通信，ful 接收来自加热炉的板坯出炉数据，通过 TAO 调用相关函数，触发调度进程 papRcv。接口进程 hmil 负责与 HMI 服务器进行通信，hmil 接收操作工在 HMI 操作界面上设定的参数数据，通过 TAO 调用相关函数，触发调度进程 papRcv。接口进程 accl 负责与 ACC（快速冷却控制仪）二级服务器通信，accl 接收来自 ACC 发送的报文，通过 TAO 调用相关函数，触发调度进程 papRcv。进程 pgl 负责与测厚仪二级服务器通信，pgl 接收来自测厚仪关于钢板测量厚度的数据，通过 TAO 调用相关函数，触发调度进程 papRcv。

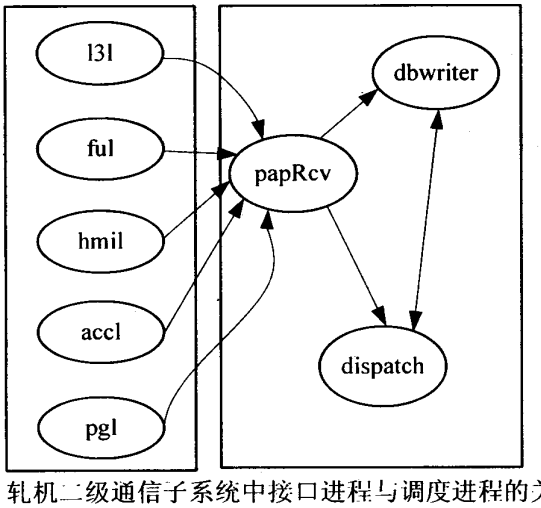

图 3.3 轧机二级通信子系统中接口进程与调度进程的关系 1

Fig. 3.3 The relationship 1 between interface process and dispatch process in Mill 2 communication subsystem

调度进程 papRcv 根据上述接口进程发送的报文和过程自动化的要求，触发调度进

程 dbwriter，将需要存储的报文数据存放到数据库中，同时触发核心调度进程 dispatch，以完成其他进程的调度。

（2）如图 3.4 所示，进程 pgl 负责与测厚仪二级服务器通信，进程 oll 是轧机二级与向轧机一级发送报文的借口，进程 spt 主要完成设定值的发送。进程 spt 可以由进程 dispatch 触发，以保证在适当的时间发送设定值。dispatch 会在一定的触发时机，例如钢坯出炉、dispatch 收到一级请求、钢板每个轧制道次后等，触发进程 spt 向进程 oll 和进程 pgl 发送设定值数据，这些设定值数据大多来自计算模型。

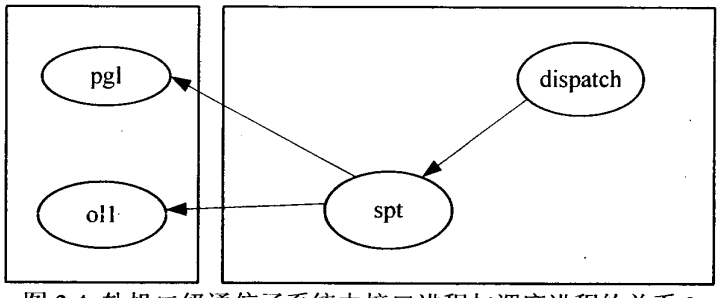

图 3.4 轧机二级通信子系统中接口进程与调度进程的关系 2

Fig. 3.4 The relationship 2 between interface process and dispatch process in Mill 2 communication subsystem

（3）如图 3.5 所示，进程 ill 是轧机一级向轧机二级收发报文的通信的接口，进程 meas 主要负责测量值的处理，meas 不需要与其他任何二级服务器（如测厚仪，冷却仪器等）进程通信。通过一定的循环监测方法，ill 将来自基础自动化的各种测量数据发送给 meas，meas 进程对这些测量数据进行收集和处理，存放在共享内存的一块特定区域，并设置开关，例如：如果开关状态为 on，则进行数据的接收，如果开关状态为 off 则停止接收数据，对已接收数据进行处理，最终将完成处理的数据发送个计算模型，进而触发 dispatch 对其他进程的调度。

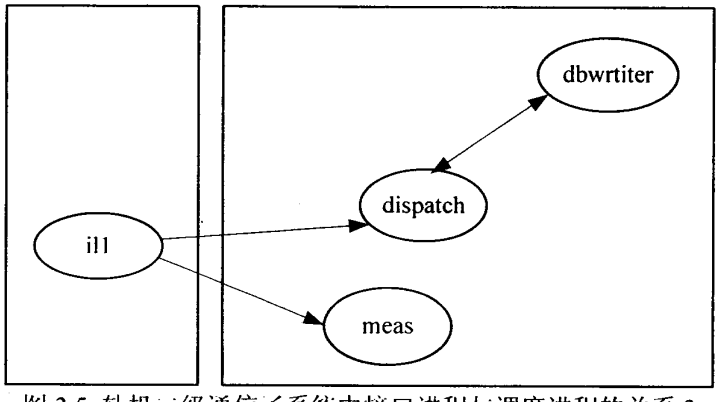

图 3.5 轧机二级通信子系统中接口进程与调度进程的关系 3

Fig. 3.5 The relationship 3 between interface process and dispatch process in Mill 2 communication subsystem

3.3.1.2 调度进程与计算模型的通信设计

本文 3.3.1.1 小节已经介绍了几种主要的调度进程以及它们与接口进程间的通信关系，这一节将重点介绍计算模型进程,并对其与调度进程的通信进行详细设计。计算模型进程主要有 3 个：precalc、recalc、和 adapt。主要用于设定值计算。

调度进程与计算模型进程有着密切的通信关系，主要有以下几种形式：

（1）如图 3.6 所示，进程 precalc 主要负责道次规程的预计算，计算指定钢板所要完成的道次规程，precalc 由 dispatch 进行触发，常见的触发条件有三种：

（a）板坯进入轧机前，测温仪将板坯最高温度、最低温度、平均温度等数据通过进程 il1 发送到轧机二级，通过 dispatch 触发 precalc，precalc 根据实测温度对轧制规程进行计算，可提高轧制规程计算的精度。

（b）当板坯经过测厚仪，测厚仪将板坯的中心厚度，边缘厚度、平均厚度等数据通过进程 il1 发送到轧机二级，通过 dispatch 触发 precalc，precalc 根据实测的板坯厚度进程规程计算，提高计算的准确性。

（c）经过几个道次后，操作工可通过人机界面手动的触发 precalc 进程，对当前钢板状态进行规程计算，然后将设定值重新发送的轧机一级，这样可以防止因意外情况导致前期预计算结果误差较大等问题。

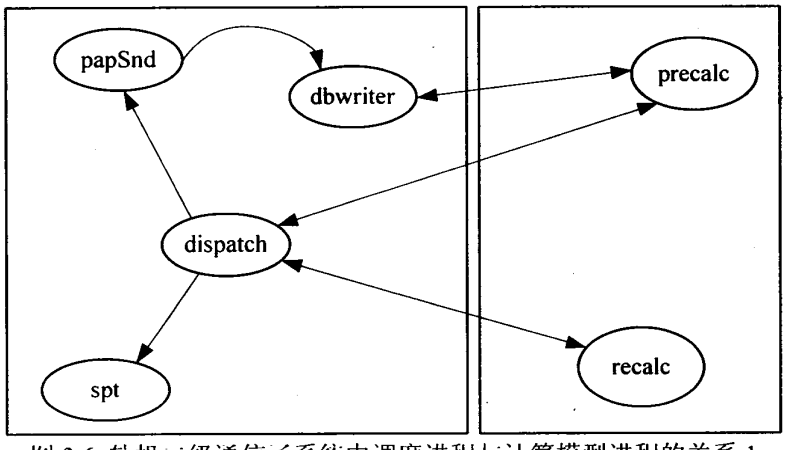

图 3.6 轧机二级通信子系统中调度进程与计算模型进程的关系 1

Fig. 3.6 The relationship 1 between dispatch process and computation model process in Mill 2 communication subsystem

进程 precalc 是计算模型进程中的核心进程，其计算的准确性对整个轧机二级系统骑着举足轻重的作用。

进程 recalc 主要负责对轧制过程中的钢板进行道次规程修正。钢板经过几个道次的轧制后，实际测量数据可能会和预计算的计算数据有所变差，为了保证最终轧制的准确性，dispatch 会在轧制过程中，根据轧机上各个仪表的实测数据触发 recalc，在原来到此

规程设定值的基础上，重新计算剩余未扎道次的设定值，并将计算结果放入共享内存，刷新预计算中的设定值数据，同时触发 dispatch 调用其他进程，将设定值重新发送到轧机一级。

（2）如图 3.7 所示，进程 adapt 主要负责对钢板轧制状态的自学习，每块钢板轧制完成后，都会在最后一个道次触发 dispatch 进程，并由它调度 adapt 进行模型的自适应计算，adapt 接收 meas 进程传送的处理过的实测数据，并比较实测数据与规程计算的数据的差异，进行模型的长期自学习。另外，adapt 还可以利用轧机二级自学习子系统提供的数据分析结果，进行长期的自学习，提高模型计算的准确性。

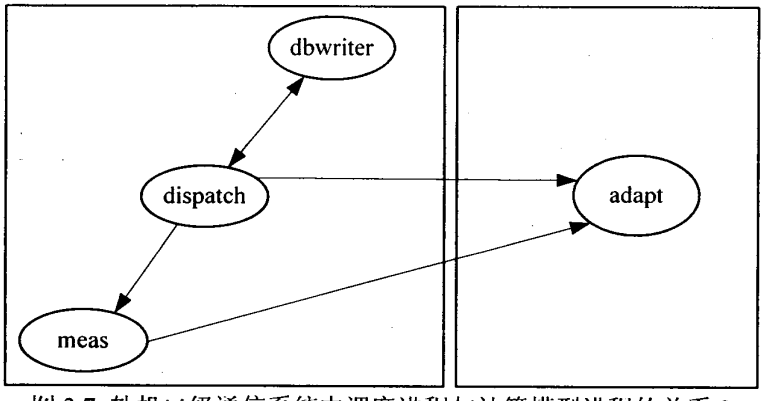

图 3.7 轧机二级通信系统中调度进程与计算模型进程的关系 2

Fig. 3.7 The relationship 2 between dispatch process and computation model process in Mill 2 communication subsystem

3.3.1.3 多进程控制序列的设计

轧机二级通信子系统采用了多进程通信的方式传送数据，其中某些进程的运行状态可能会影响到其他进程任务的正确执行，例如所有接口进程能正确收发数据的必要条件是 dispatch 进程已启动并正常运行，spt 进程能正确向一级发送数据的前提条件是 ol1 进程已正常运行，precalc 进程能正确完成计算任务并将计算结果返回二级的必要条件是 dbwriter 进程和 dispatch 进程的运行状态正常，因此必须采用一定的控制方法来解决各个进程的这种相互依赖关系，确保系统中多进程通信的正确性。

本文提出的多进程控制序列，其目的就是为了找到各个进程间的相互依赖关系，并利用一定的技术手段对这种依赖关系进行存储并加以利用，以保证每个进程任务都能正确的执行，从而解决轧机二级通信子系统中多进程通信的正确性问题。以下是多进程控制序列的具体设计步骤：

（1）在设计控制序列之前，首先需要由轧机二级通信子系统中的核心进程 dispatch 创建一个控制矩阵，存放于共享内存中，并由它进行维护。

（2）每个进程在根据自己当前的运行状态实时的将状态信息填入控制矩阵中，每

个进程可以有三种状态：0 空闲、1 忙、2 异常。

（3）控制序列和决策集的对应以控制矩阵的行为单位，如表 3.3 所示，进程 1 的控制序列为 1010101，表示任务进程 1 的正确执行需要依赖进程 3、进程 5、进程 7，进程 2 的控制序列为 0101001，表示任务进程 2 的正确执行需要依赖进程 4、进程 7，进程 3 的控制序列为 0010100，表示任务进程 3 的正确执行只依赖进程 5 的运行状态，以此类推，每个进程都会有一个相应的进程控制序列。当某个进程（以进程 1 为例说明）向 dispatch 进程发出任务请求，dispatch 进程会根据控制矩阵找到进程 1 的控制序列，并判断进程 1 的控制序列中所依赖的其他进程（进程 3、进程 5、进程 7）的运行状态是否正常，如果进程 3、进程 5、进程 7 都处于非异常状态，则 dispatch 允许任务进程 1 执行，如果进程 3、进程 5、进程 7 中有异常状态，则 dispatch 会重新启动所有异常的进程，并根据控制序列再一次判断，直到满足任务进程正确执行的条件。

表 3.3 多进程通信控制矩阵

Table 3.3 The control matrix of multiprocess communication

	进程 1	进程 2	进程 3	进程 4	进程 5	进程 6	进程 7
进程 1	1	0	1	0	1	0	1
进程 2	0	1	0	1	0	0	1
进程 3	0	0	1	0	1	0	0
进程 4	1	0	0	1	0	1	0
进程 5	0	1	0	0	1	0	1
进程 6	1	0	1	0	0	1	0
进程 7	0	1	0	1	0	0	1

3.3.1.4 数据库的设计

在实际的中厚板生产线上，整个轧机二级通信子系统需要完成复杂的数据通信任务，这种生产任务可能是全天候的，系统需要大量的生产数据为生产过程提供数据基础，同时也会产生大量的计算结果数据，对生产过程进行修正，所以需要利用数据库对生产中的数据进行存储。由于数据库读写操作频繁，涉及的数据量大，因此结合实际的生产需要，在对现有的工业自动化广泛应用的主流数据库进行技术考察后，本课题选取了 Oracle 10g 作为轧机二级通信子系统的数据库。

Oracle 10g 是目前最流行、最强大的数据库系统，具有完备的数据管理功能，能完美的刻画数据关系，并实现了完善的分布式处理功能[42]。

与其他数据库（如 SQL Server、MySQL）不同，Oracle 10g 的下一层逻辑结构不是数据表，而是表空间，每一个数据表都属于唯一的表空间，表空间是 Oracle 的开创性理

念，使得数据库管理更加灵活，极大的提高了数据库的性能。与数据库、数据表类似，表空间也是一个逻辑对象，它与数据库和数据表的关系如图 3.8 所示。

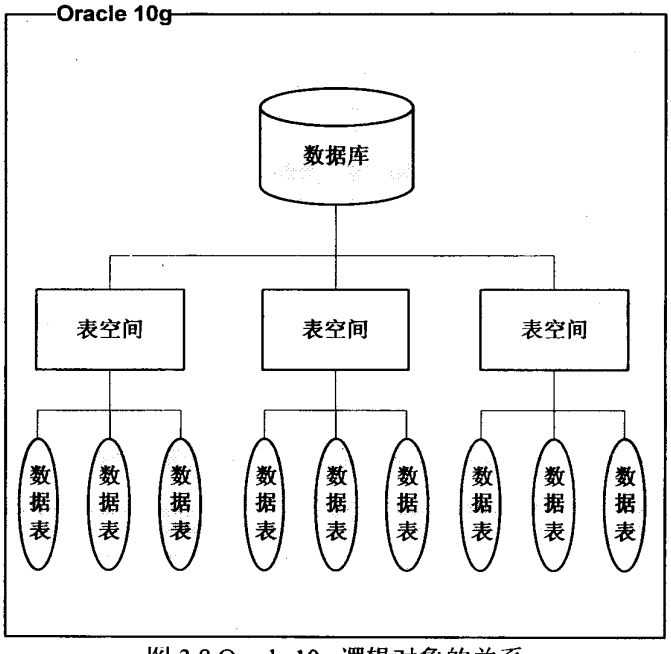

图 3.8 Oracle 10g 逻辑对象的关系
Fig. 3.8 The relationship of Oracle 10g logical objects

从图中可以看出一个数据库可以有多个表空间，而一个表空间也可以有多个数据表，表空间的使用是 Oracle 10g 高性能的保证，具体有以下四个优点：

（1）避免磁盘空间突然耗竭的风险

在创建表空间时，需要预先指定表空间的最大磁盘空间，从而避免数据库中数据的无限扩张，造成磁盘突然耗竭。

（2）规划数据灵活

当数据库中的数据量很大时，可以根据需要自定义划分标准，将数据表划分到表空间中，这种类似操作系统文件夹的功能可以方便数据的查询和管理。

（3）提高数据库性能

对于频繁访问的数据表，可以将其放入单独的表空间，并存储于高性能磁盘；同时将访问较少的数据表也放入单独的表空间，并存储于性能较低的磁盘。这样可以合理的利用硬件资源，最大限度的提高数据库性能。

（4）提高数据库安全性

不同的表空间对应着不同的物理文件，当某个表空间的物理文件被损坏时，不会影响其他表空间的正常使用，提高了数据库的安全性。

利用 Oracle 10g 自带的 Database Configuration Assistant 创建轧机二级通信子系统需要的数据库 NEURollDB，再根据需要创建表空间 ROLL_TABLESPACE，然后再根据需要创建不同的数据表，分别存储轧机二级通信子系统中的数据信息，例如创建表 PID 用来设定轧制过程中的主要参数，创建表 LOG 用来存储系统中各个进程的日志，创建表 ROLL_SEQUENCE 用来存储轧制序列等。

3.3.2 轧机二级监控子系统的设计

轧机二级通信具有复杂性、实时性等特点，为了保证轧机二级各个进程能够正确的运行，必须对整个系统进行监控，实时的反应系统当前的运行状态，并能够对系统中的重要进程进行一定的控制。Windows 窗体应用程序提供了大量的窗体设计控件，利用这些控件设计监视界面简单方便，本节主要对轧机二级监控子系统进行详细设计，具体分为监控界面的设计和事件响应的设计。

3.3.2.1 监控界面的设计

轧机二级监控子系统的监控界面根据功能划分主要有两大部分：进程监控和数据监控。

（1）进程监控。轧机二级通信子系统中的各个进程运行状态主要分为三种：正常运行、异常运行和停止运行。利用 Windows 窗体应用程序中的 textBox 控件背景颜色的改变可以对进程的状态进行标识，例如正常运行时，进程状态为绿色，异常运行时进程状态为黄色，停止运行时，进程状态为红色。通过颜色的变化，实时的反应系统中各个进程的运行状态，在遇到异常情况时能够让工作人员及时的做出调整。

为了及时方便的了解系统中各个进程的通信状况，在进程监控部分还设计了系统运行信息模块，通过对共享内存的访问实时的反应系统中各个进程间通信的状态，其中包含的主要字段有：通信时间、通信进程名以及简单的通信报文信息。通过系统运行信息的实时刷新，工作人员可以方便的了解到当前系统中进程的通信情况，便于对整个轧制过程进行控制。

（2）数据监控。数据监控主要是对数据库中存储的数据信息进行监控，其中包括对数据表的查询和对系统日志的查询。对数据表的查询主要方便工作人员查询钢板或者板坯的固有参数，了解更多的钢板信息。系统日志查询主要为工作人员提供系统在某一段时间内进程通信情况的记录，如果出现问题，利用系统日志可以方便的观察系统运行记录，从而为工作人员能够顺利发现问题、解决问题提供帮助。

dataGridView 控件是 windows 窗体应用程序中自带的、用于显示和编辑多种不同类型数据源的表格数据的工具。dataGridView 具有很高的可配置性和可扩展性，提供大量

的属性、方法和事件。在数据监控中，数据表格和系统运行日志中包含大量数据，利用dataGridView控件进行数据的显示，不但可以方便于编程，而且还可以减小内存开销。

3.3.2.2 事件响应机制的设计

轧机二级监控子系统的目的是为了实时方便的了解轧机二级中各个进程的通信状态以及它们的通信关系，并根据实际情况进行一定的控制操作。本文3.3.2.1小节对监控界面做了详细的设计，但是要想完成整个监控系统必须依靠事件响应机制给予支持，才能实现监控的功能。

针对监控界面的不同控件，主要有以下几种事件响应方式：

（1）脚本调用。进程启动按钮是Windows窗体应用程序中的一个button控件，通过对进程启动按钮的点击，会调用一个外部脚本，全面开启轧机二级系统中的各个进程，这些进程的启动顺序是有一定的先后关系的，例如dispatch是守候进程，必须最先启动等，利用点击按钮调用外部脚本的事件响应机制可以很好的控制进程启动顺序，另外如果需要对进程启动方案进行调整，只需要对脚本进行简单的修改就可完成。

（2）函数调用。监控界面中的textBox控件和控制每个进程的button控件都是用来监控它们对应进程的运行状态。通过对button按钮的点击，可以调用系统函数或者自定义函数进行事件的响应，实现每个进程运行状态的单独改变，同时调用自定义函数改变textBox控件的背景颜色，使得对进程的监控更加方便。

（3）读取数据库。dataGridView 是专门用来显示数据的控件，通过选择监控界面上不同的数据表，dataGridView进行事件响应，绑定相应的datasource数据源，同时根据条件对数据库进行查询和读取，将数据结果显示到监控界面上来。

3.3.3 轧机二级自学习子系统的设计

轧机二级自学习子系统主要功能是利用实测数据对模型进行自适应，提高模型计算精度。本节利用 DW+OLAP 框架结构，结合生产线的实际情况，对轧机二级子系统进行详细设计。

3.3.3.1 数据仓库的设计

数据仓库是面向主题的、集成的、包含历史的，依照数据仓库的特点，数据仓库的设计主要包括了数据源的选择、数据仓库的主题设计、模型选择以及对源数据的抽取、转换、装载设计等几大部分。

数据仓库的数据源种类是多样的，如Oracle 8/9i/10g/11g系列的数据源、操作型数据库数据源、或是以Excel、TXT等文件格式存储的数据源等。本文所用到的数据源都是由东北大学轧制技术及连轧自动化国家重点实验室提供，这些数据源主要包括了板坯

固有属性、板坯轧制温度测量值、轧制力控制、轧制时间记录等信息。

数据仓库是面向主题的，主题是一个抽象的概念，每一个主题是决策这所关心的问题。轧制力在规程计算过程中是一个重要的计算对象，本文选择了轧制力为一个主题，考察对于不同类型的钢板在轧制过程中对轧制力的需求情况。

在确定了主题之后，要根据主题对数据仓库系统进行建模，在数据仓库系统中常使用的建模方法为维度建模，模型包括事实表和维度表，事实表不允许改变，而维度表可以进行修改。维度建模又可以细分为星型模型和雪花模型两种，星型模型以事实表为中心，描述主题的数据，并与维度表相连，雪花模型是星型模型的扩展，比星型模型更加复杂。本文以轧制力为主题，根据实际需要选择建立如图 3.9 的星型模型，通过这个星型模型可以了解某段时间的某种钢板对于不同类别温度的轧制力信息。

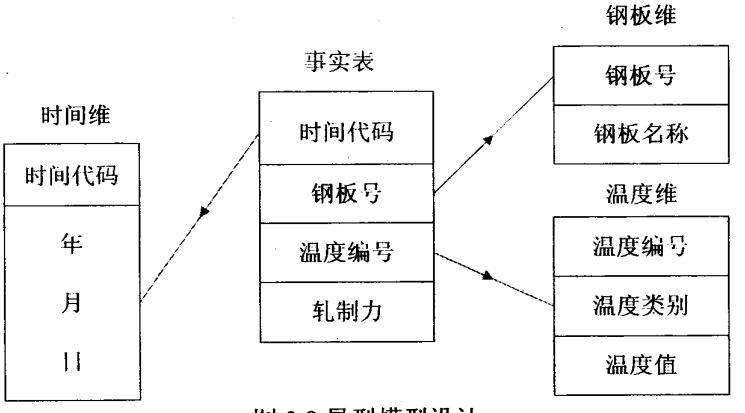

图 3.9 星型模型设计

Fig. 3.9 The star model designing

数据仓库中的数据可能来自于不同的数据源，这些数据不能直接的进入数据仓库，需要根据已经确定的主题对数据进行抽取，经过清洗处理转换后，最终装载到数据仓库。ETL 主要完成数据的抽取、转换和装载，其工作流程如图 3.10 所示。

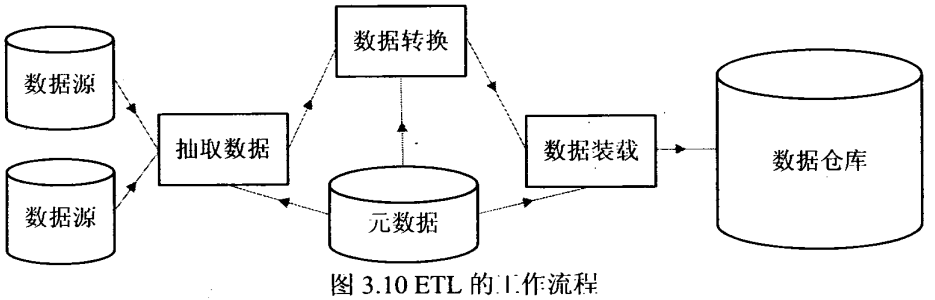

图 3.10 ETL 的工作流程

Fig. 3.10 The workflow of ETL

数据的抽取主要是对数据源中与主题相关的数据进行提取，丢弃与主题无关的数据，以及那些无效的坏数据；数据转换将来自不同数据源中不同格式的数据进行转换处理，成为数据仓库可识别的唯一的数据格式，数据装载主要负责将经过抽取和转换的数

据装载到数据仓库中，其中抽取和转换是 ETL 过程中比较重要的两部分，而数据装载是一个相对比较机械的过程[43]。

3.3.3.2 OLAP 的设计

OLAP（联机分析处理）主要用于信息的检索，动态的访问数据仓库中的数据。OLAP 主要包括了以下几个分析方法：钻取、切片、切块、旋转等[44]，利用 OLAP 技术，用户能够从多个侧面、多个维度查询数据，从而了解数据背后蕴含的规律。

（1）钻取

钻取又分为上钻和下钻，上钻是从相对具体的数据出发，去考察与它相关的相对概括的数据查询，而下钻正好相反是从相对概括的数据出发，去考察与其相关的相对具体的数据的查询。比如在时间维度中，从“年”数据出发去分析“月”、“日”的数据就是下钻，反之就是上钻。在实际分析中，可以根据需要选择适当的钻取深度。

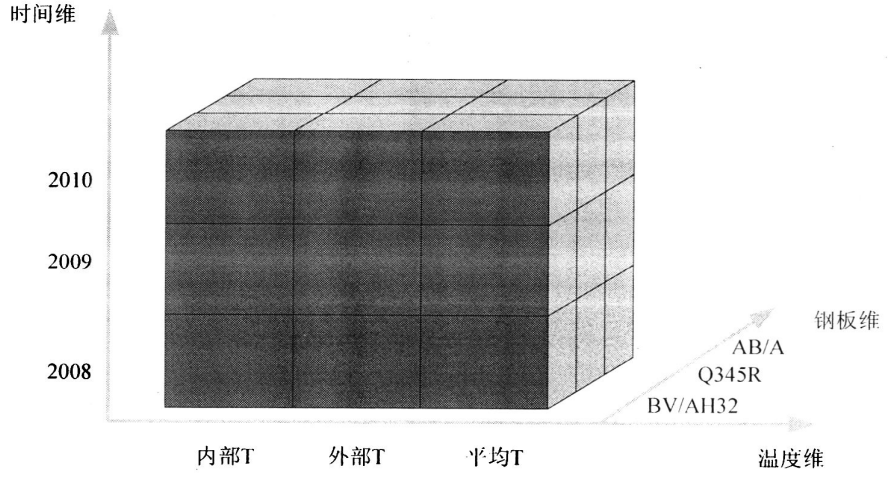

图 3.11 多维数据库数据立方体

Fig. 3.11 Data cube of multidimensional database

（2）切片和切块

切片和切块是在确定一部分维的选定值后，考察其他维上度量数据的分布。例如，在图 3.11 给定的多维数据库数据立方体中，如果选择时间维和钢板维的所有维成员，分别考察温度维中的内部温度、外部温度和平均温度所对应的轧制力水平，就是在按照温度维进行切片，如图 3.12 所示。

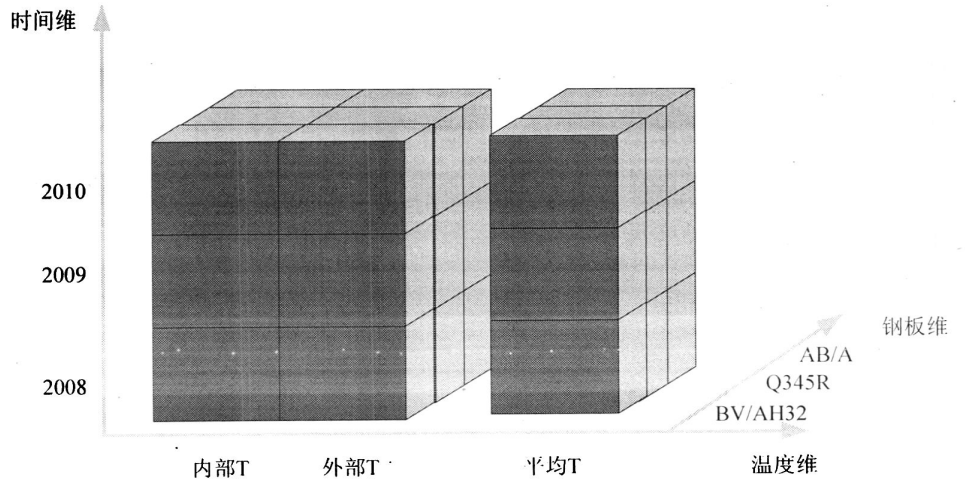

图 3.12 数据立方体的切片
Fig. 3.12 Data cube slicing

切块可以看成几个切片的叠加，图 3.13 给出了按照时间维、温度维和钢板维对数据立方体的切块，如时间维选择 2009 年、2010 年，温度维选择外部温度和平均温度，钢板维选择 BV/AH32、Q345R。

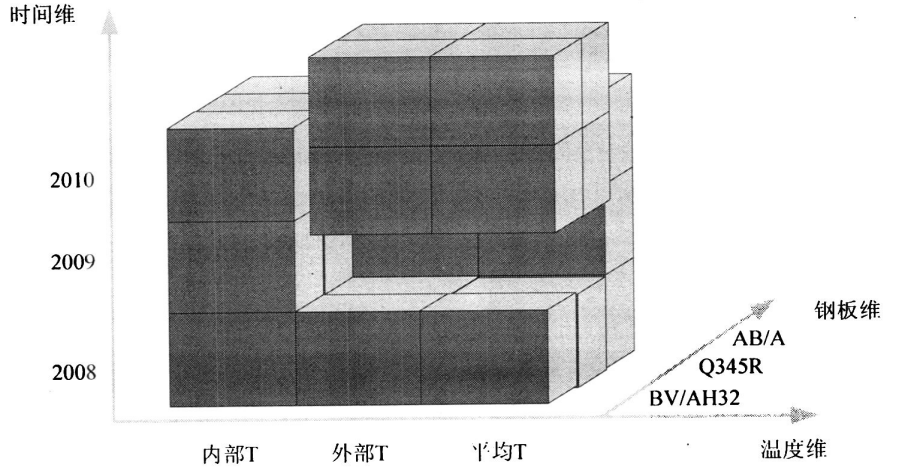

图 3.13 数据立方体的切块
Fig. 3.13 Data cube slicing

（3）旋转

旋转是改变维的方向，通过重新安排维的位置而得到不同的数据，包括行维和列维的交换，页面内的维和页面外的维交换等，例如将钢板维固定，交换时间维与温度维，或者将温度维固定，交换钢板维和时间维等。

除了以上三种比较典型的 OLAP 操作外，还有许多其他操作，例如取线、取点、钻透等，本文没有涉及到这些比较复杂的 OLAP 操作，在此也不再做进一步的分析。

3.4 本章小结

本章根据中厚板生产线的实际生产情况，对中厚板轧机二级通信平台调度监控系统进行了全面的需求分析，在需求分析的基础上基于 ACE、TAO、OLAP、OO4O 等技术对系统进行了概要设计，对系统中各个进程的通信方式和数据库访问方式进行了讨论，最后根据概要设计针对系统中的各个子系统给出了详细的设计方案。

第4章 通信调度监控系统的实现

本章根据本文第三章对中厚板轧机二级通信平台调度监控系统的概要设计与详细设计，得益于 ACE、TAO、OO4O、OLAP 等关键技术的支持，完成了系统的实现。本章主要介绍了系统实现过程中的重要部分，包括轧机二级通信子系统、监控子系统和自学习子系统的具体实现过程和方法，并讨论了系统开发平台的选型和开发环境的配置问题。

4.1 开发平台的选型

在系统进行实际开发前，课题组对国内几家大型钢铁企业目前的生产线状况进行了调研，并实地考察了国内某企业的中厚板生产流程以及生产线上使用的硬件环境，结合目前国内相关方面主流产品的使用和课题组的实际情况，提出了如下的平台选型：

（1） 操作系统：Windows Xp

（2） 开发环境：Visual Studio 2008（VC9.0）

（3） 中间件：ACE（version 5.7, released Jun 22 2009）、TAO（version 1.7, released Jun 22 2009）

（4） 数据库：Oracle 10g

（5） 其他工具软件：OWB_11.2.0.1_Windows、PLSQL Developer（version 7.1）、ActivePerl 5.8.8 Build 822

本文前面已经对 ACE、TAO 和 Oracle 10g 进行了详细的介绍，讨论了它们的组织结构以及各自的优点，在此不再赘述。

Oracle Warehouse Builder (OWB)是 Oracle 的一个综合工具[45]，帮助用户设计、部署与管理数据仓库，是 Oracle 用于设计与部署数据仓库解决方案的技术，它提供对数据仓库的 ETL（抽取、转换和加载）、完全集成的关系和维度建模、数据质量、数据审计等功能，另外还可以对数据和元数据进行生命周期的管理，为设计、部署企业数据仓库数提供便利。

使用 PL/SQL Developer 是为了方便、快速地设计 Oracle 10g 数据库而选用的。PL/SQL Developer 是一个集成开发环境，专门面向 Oracle 数据库存储单元的开发。PL/SQL Developer 侧重于易用性、代码品质和生产力，能够使得 Oracle 应用程序的开发变得方便灵活。

由于在 win32 的平台上没有启动 perl 脚本的方法，所以本文选用了 ActivePerl 工具，

ActivePerl 是一个可以执行 Perl 程序的工具软件。其包括：Perl for Win32、Perl for ISAPI、PerlScript、Perl Package Manager 四套开发工具程序，可以编写出适用于 unix，windows，linux 系统的程序。

4.2 轧机二级通信子系统的实现

本节根据 3.3.1 节提出的轧机二级通信子系统的详细设计方案，运用 ACE、TAO 等关键技术，在已选的开发平台下实现了轧机二级通信子系统。本节重点介绍了轧机二级通信子系统中的几个典型的、有代表性的进程的实现方法，包括守候进程、接收进程、发送进程和预计算进程，通过这些进程的设计实现，解决整个轧机二级通信子系统的技术难题。

4.2.1 接收进程的实现

接收进程 papRcv 是负责与接口进程进行通信，例如接收 pgl 进程（来自测厚仪的数据）、l31 进程（来自生产管理级的参数配置数据）发送来的上行报文，然后触发 dispatch 进程将报文数据写入共享内存或者数据库，图 4.1 给出了 papRcv 进程的交互图。

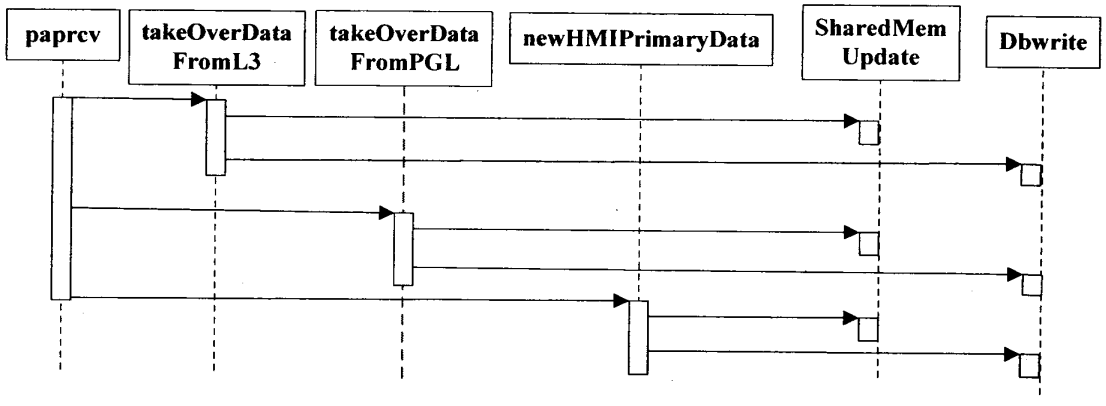

图 4.1 paprcv 进程的交互图
Fig.4.1 The interaction diagram of papRcv process

为了保证轧机二级通信子系统的实时性，papRcv 进程必须能够快速的处理来自接口进程的报文。TAO 是一种开源的 CORBA 实现，并通过自适应通信环境(ACE)提供的软件概念和框架构建而成，通过定义标准的.idl 接口文件，生成用于服务器和客户端的静态存根和框架，每个服务器端可以对应多个客户端，函数的具体功能都在服务器端实现，客户端只需简单的包含指定的头文件就可对在.idl 中声明的函数进行调用，实现快速的响应，这里以 papRcv 进程为例，具体描述一下利用 TAO 的 ORB 机制对 papRcv 进程的实现。

如图 4.2 所示，papRcv 进程的.idl 文件中定义了三个函数分别用于与接口进程进行通信，利用 TAO 自带的编译器 tao_idl 编译定义的.idl 文件，可以自动生成服务器端、客户端的.h 和.cpp 文件，具体命令为：tao_idl –GI ./paprcv.idl。

```
#if !defined (_PAPRCV_IDL)
#define _PAPRCV_IDL
#include "SequenceTypesIDL.idl"
Interface PapRcvIDL
{
    //telegram from L3
    oneway void takeOverDataFromL3 ( in OctetSequence seq);

    //telegram from PGL
    oneway void takeOverDataFromPGL ( in OctetSequence seq);

    //new primary data set via HMI
    oneway void newHMIPrimaryData ( );
}
```

图 4.2 papRcv 进程的 IDL 文件
Fig.4.2 IDL file of papRcv process

TAO 提供了一个 make 应用程序，通过它可以跨平台的管理项目的创建和编译，为了将通过 tao_idl 编译生成的文件包含在一个工程中（papRcv 工程），可以编写 papRcv 进程的.mpc 文件，定义项目创建所需的源代码，并通过命令：mpc.pl -type vc9 ./paprcv.mpc 生成工程文件。最后通过命令：mwc.pl -type vc9 ./paprcv.mwc，在 windows 上生成最终的 solution 文件，具体实现过程如图 4.3 所示。

```
D:\HowToTAO>tao_idl -GI paprcv.idl
tao-idli_jkSk3T.cpp
processing paprcv.idl

D:\HowToTAO>mpc.pl -type vc9 ./paprcv.mpc
Using D:/ACE_wrappers/bin/MakeProjectCreator/config/MPC.cfg
CIAO_ROOT was used in the configuration file, but was not defined.
DDS_ROOT was used in the configuration file, but was not defined.
Generating 'vc9' output using paprcv.mpc
Generation Time: 1s

D:\HowToTAO>mwc.pl -type vc9 ./paprcv.mwc
Using D:/ACE_wrappers/bin/MakeProjectCreator/config/MPC.cfg
CIAO_ROOT was used in the configuration file, but was not defined.
DDS_ROOT was used in the configuration file, but was not defined.
Generating 'vc9' output using paprcv.mwc
Generation Time: 1s

D:\HowToTAO>_
```

图 4.3 papRcv 进程 的 VC solution 生成过程
Fig.4.3 The process of VC solution created of papRcv process

通过 Visual Studio 2008 打开之前生成的 solution 文件，可以看到其中包含的三个工程文件：papRcv 服务器端文件、papRcv 客户端文件和 papRcv 接口文件，接口文件主要对在通信过程中需要使用的函数进行声明，服务器端的文件主要负责对接口文件中定义的函数进行具体的功能实现，它也是 papRcv 进程所在的工程，在其他进程与 papRcv 进

程通信时，如果需要调用相关的 TAO 函数，只需将工作区中客户端工程中的 papRcvC.h 头文件包含进去，便可直接对相关 TAO 函数调用，并不需要了解函数的具体实现过程，这样以来函数的功能完全由服务器端进行统一控制，客户端在需要时进行简单的函数调用即可，这大大简化了各个进程间的通信过程，保证了轧机二级通信子系统的实时性，并提供了分布式的通信服务。

这种基于 ORB 的通信方式，为整个轧机二级通信的分布式部署带来便利，同时也存在一些问题，在生成的服务器端和客户端中，可能存在一个服务器对应多个客户端的情况，正如轧机二级通信子系统中的 papRcv 进程，pgl、l3l、himl 等都是它的客户端，这样以来如何对不同客户端进程进行识别，成为一个关键问题。TAO 提供了命名服务功能，具体的实现过程已经封装在相关的头文件中，在开发过程中只需包含相关头文件，通过 bind 和 find 方法就可实现命名服务。bind 方法主要为不同对象进行命名，将名字与不同对象进行绑定，形成一种映射关系，每一个名字对应一个对象引用，但是一个对象引用却可以对应多个名字，bind 的具体实现方法如图 4.4 所示。

```
ORBObjectSet::mill2_object_var paprcv =  mill2_object_i._this();
CORBA::Object_var naming_context_object = orb->resolve_initial_references ("NameService");
CosNaming::NamingContext_var naming_context = CosNaming::NamingContext::_narrow (naming_context_object.in ());
CosNaming::Name name (1);
name.length (1);
name[0].id = CORBA::string_dup ("paprcvIDL");
naming_context->bind (name, mill2_object.in ());
```

图 4.4 对象命名过程

Fig.4.4 The naming process of an object

find 方法是对名字的解析，给定一个名字就可以找到该名字的对象引用，这样不需要了解对象的物理位置，只根据名字就可以获取对象的引用，实现了分布式环境中的网络透明。find 的具体实现方法如图 4.5 所示。

```
CORBA::Object_var naming_context_object =  orb->resolve_initial_references ("NameService");
CosNaming::NamingContext_var naming_context = CosNaming::NamingContext::_narrow (naming_context_object.in ());
CosNaming::Name name (1);
name.length (1);
name[0].id = CORBA::string_dup ("paprcvIDL");
CORBA::Object_var factory_object = naming_context->resolve (name);
ORBObjectSet::mill2_object_var paprcv = ORBObjectSet::mill2_object::_narrow (mill2_object.in ());
```

图 4.5 对象名字解析过程

Fig.4.5 The name resolvling process of an object

papRcv 进程接收来自接口进程的报文数据，经过处理后会将数据存放到共享内存池对共享内存池进行更新操作，以便保证其它进程的协同计算的准确性和实时性，或者存放于数据库进行永久性存储。具体的存放规则如下：

（1） 需要经常更新的数据要存放在共享内存池中，否则存放在数据库中。

（2） 需要被两个或者两个以上进程使用的数据要存放在共享内存中，加快进程查找使用数据的速度，避免频繁的数据库操作。

（3） 控制进程通信的互斥、同步变量要放在共享内存中，从而对系统中各个进程间的通信进行实时的控制。

（4） 具有历史性的、长期性的、稳定的数据信息要存放于数据库，实现数据的永久性存储。

共享内存池是由 dispatch 进程负责创建和维护的，papRcv 进程只需要在一定条件下调用 find 方法，便可对共享内存池进行访问。find 方法对共享内存块的名字进行解析，可以简单、快速的定位所要访问的共享内存块，papRcv 进程访问共享内存的具体实现方法如图 4.6 所示。

```
typedef ACE_Malloc<ACE_MMAP_MEMORY_POOL, ACE_SYNCH_MUTEX> malloc_mmap;
ACE_MMAP_Memory_Pool_Options options(ACE_DEFAULT_BASE_ADDR,
ACE_MMAP_Memory_Pool_Options::ALWAYS_FIXED );
malloc_mmap malloc_mmap_t(BACKING_STORE,BACKING_STORE, &options);
malloc_mmap *g_malloc_mmap = &malloc_mmap_t ;
int ACE_TMAIN(int argc, ACE_TCHAR *argv[])
{
        ......
        if(g_malloc_mmap->find("PAPRCV_SHM_MESSAGE",P_OpInput) == -1)
        {
                ACE_ERROR_RETURN((LM_ERROR,ACE_TEXT("%p\n"),ACE_TEXT("PAPRCV: No Data")),-
1);
                ACE_OS::exit(1);
        ACE_Time_Value time_(4);
        ACE_Reactor::instance()->schedule_timer(new timer_handler, NULL, time_, time_);
        while (true)
        {
                ACE_Reactor::instance()->handle_events();
        }
        ......
}
```

图图 4.6 papRcv 进程访问共享内存池的实现

Fig.4.6 The implementationof papRcv process accessing shared memory pool

对于需要永久性存储的数据，papRcv 进程将它们存放于 Oracle 10g 数据库中，具体的实现采用的是 OO4O 数据库访问技术，根据本文 2.5 小结对 OO4O 技术的介绍，初步了解了 OO4O 的几个主要类库，如 ODatabase 类、ODynaset 类、OField 等，利用这些类库可以实现对数据库的快速访问，从数据库连接、数据库记录集的创建管理直到对每个数据表的字段的操作都可以通过 OO4O 自带的 C++类库完成，在具体的实现过程中，需要包含 oracle.h 头文件和 ORACLEM32.lib 文件。papRcv 进程利用 OO4O 技术访问数据库的具体实现如图 4.7 所示。

```
#include "oracl.h"
.....
Odatabase m_odb;
if (!m_odb.IsOpen())
{
    try{
            Ostartup();
            m_odb.Open("neural","scott","neuralmill2");
      }catch (OException E) {ACE_OS::printf("open error\n");}
}
ACE_OS::printf("paprcv connects to Oracle HMI_INPUT\n");
try{
          Cstring sql = "select * from HMI_INPUT";
          ODynaset rs1(m_odb,sql);
          if(OSUCCESS!=rs1.ADDNewRecord())
              return -1;
          rs1.SetFieldValue(0,OpInput_t->Plate_ID);
          .....
          rs1.Update();
   }catch (OException E) {ACE_OS::printf("add error\n");}
```

图 4.7 papRcv 进程通过 OO4O 访问 Oracle 数据库的实现

Fig.4.7 The implementation of papRcv process accessing Oracle database through OO4O

以上对接收进程 papRcv 的通信功能的实现进行了完整、全面、详细的介绍，阐明了 papRcv 进程与其相关参与者之间的协作关系。在接下来对其他进程的介绍中，会用到与 papRcv 进程类似的实现方法，对相似部分，本文只做简单的功能介绍，不予赘述。

4.2.2 守候进程的实现

守候进程 dispatch 是整个轧机二级通信子系统的核心进程，负责对其他全部进程进行调度管理。dispatch 是连接接口进程、控制进程和计算模型进程的纽带，是整个轧机二级通信子系统的神经中枢，主要有一下几个任务：

（1）管理轧机二级通信子系统其他全部进程的启动与关闭。

（2）创建并维护共享内存池和控制矩阵，保证进程间通信的正确性。

（3）根据接收进程 papRcv 的上行报文，对整个轧机二级通信子系统进行调度。

（4）触发发送进程 papSnd 向其他外部服务器发送下行报文。

（5）与 precalc 进程进行互操作，将收到的来自一级的实测数据传递给 precalc 进程进行计算。

为了更好的描述 dispatch 是如何完成上述任务，根据其在通信过程中与其他进程的调用关系，给出了如图 4.8 所示的 dispatch 进程的交互图。

ProcessSC 方法可以通过 ACE 提供的进程管理器 ACE_Process_Manager 来实现，ACE_Process_Manager 允许用户通过单次调用派生多个进程，直到它们终止，同时可以在实际处理器上进行登记，方便子进程结束时的回调。ACE_Process_Manager 类的 spawn_n（）方法可以一次创建多个进程，在使用之前必须先创建一个

ACE_Process_Options 对象，并把它传给 spawn_n（）实现多进程的启动和关闭，另外通过这种方法还可以强行终止之前由 ACE_Process_Manager 派生的进程。利用 ACE 的进程管理器实现多进程的启动和关闭，可以保证轧机二级通信子系统中守候进程对其他所有进程的有效管理和调度。

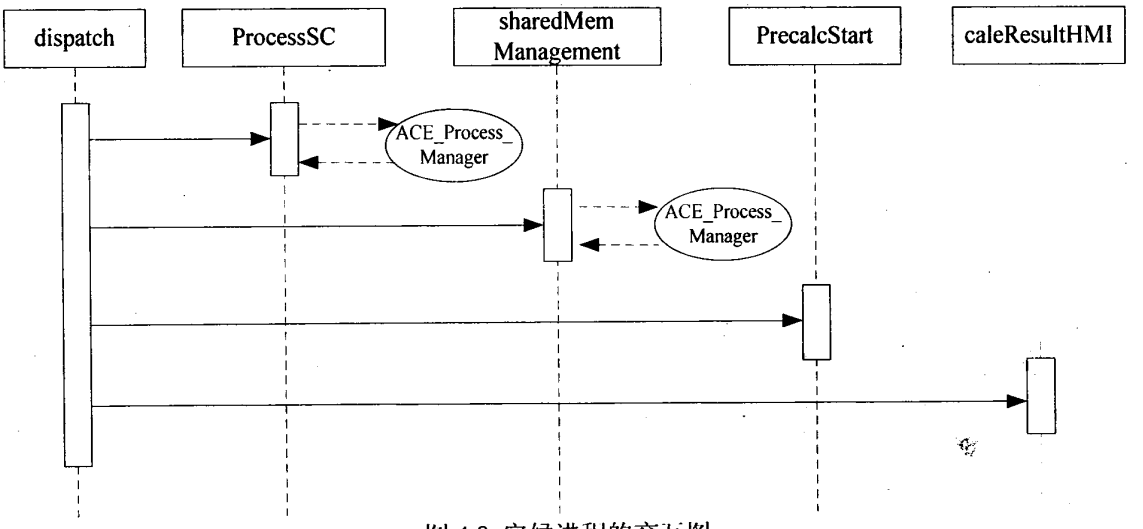

图 4.8 守候进程的交互图

Fig.4.8 The interaction diagram of waiting processes

sharedMemManagement 方法主要完成共享内存池的创建于维护。共享内存是最快的 IPC 机制，特别是在有大量数据需要共享的情况下，同时共享内存通过把文件映射进内存，还可以对文件进行处理操作，在程序运行中使得程序可以直接对内存操作而不是使用文件的 I/O 操作。ACE 提供了若干工具帮助共享内存的使用，结合本课题实际情况，守候进程 dispatch 的 SharedMemManagement 方法采用了 ACE_Malloc 创建共享内存池，内存池的类型为 ACE_MMAP_Memory_Pool。其定义方法如图 4.9 所示。

```
#define __MEMORYMALLOC_H

#include "ace/Malloc_T.h" //定义"ACE_Malloc<>"分配器类模板
#include "ace/MMAP_Memory_Pool.h" //内存池的类型，ACE_MMAP_MEMORY_POOL

#define BACKING_STORE "../backing"     //存放后备文件的地址和名字

typedef ACE_Malloc<ACE_MMAP_MEMORY_POOL, ACE_SYNCH_MUTEX> malloc_mmap;
                                        //初始化类模板的分配器族

ACE_MMAP_Memory_Pool_Optionsoptions(ACE_DEFAULT_BASE_ADDR,ACE_MMAP_Memory_Pool_Options::ALWAYS_FIXED );
                                        // 设定共享内存的起始地址

malloc_mmap malloc_mmap_t(BACKING_STORE,BACKING_STORE, &options);
```

图 4.9 共享内存池定义方法

Fig.4.9 The method of shared memory pool definition

在使用共享内存时常常面临这样一个问题，分配器可能无法对共享内存池中的所有

进程分配相同的基地址，这样以来不同基地址的进程如果需要访问其他基地址进程存放于共享内存中的数据就不会实现，无法找到存入数据所在的地址空间[46]，另外如果需要分配更多内存时，底层的内存池增长，系统可能需要把池映射到另外的基地址，造成寻址失败等问题。

ACE_MMAP_Memory_Pool_Options 允许指定很多选项，包括基地址的指定，如表 4.1 所示，这样以来在增加记录时，OS 可以把地址映射到任何地址，而在显示记录时映射到固定的地址，这样以来即使分配器每次运行时都被映射到不同的地址，程序也可以正常工作。

表 4.1 ACE_MMAP_Memory_Pool_Options 属性
Table 4.1 The options of ACE_MMAP_Memory_Pool_Options

选项名称	描述
base_address	指定一个起始地址，文件将从此开始映射进程的内存空间
use_fixed_addr	指定何时能把基地址固定下来，必须为以下三个常量之一： 1. FIRSTCALL_FIXED：在初次获取时使用指定的基地址 2. ALWAYS_FIXED：总是使用指定的基地址 3. NEVER_FIXED：总是让 OS 指定基地址
guess_on_fault	出错时尝试猜测出错地址，并重新映射/扩大所映射的文件，掩盖此问题
file_mode	在创建后备文件时使用的文件访问模式

创建共享内存时需要使用 malloc（）方法获取共享内存块的起始地址，在内存的动态存储区中分配一个指定长度的连续内存空间，并将这块连续空间的首地址返回给一个指向该内存块的指针，之后利用 bind（）方法可以为共享内存块命名，利用 find（）方法指定需要访问的共享内存块名字。这样以来各个进程便可以正常的使用共享内存池进行通信，另外 BACKING_STORE 用于备份共享内存中的数据，为异常事件处理和数据恢复提供保障。

在程序的实际运行过程中，如果某个进程修改共享内存中的数据，则需要对共享内存块进行刷新，防止其他进程访问时出现“脏读”的问题，为了能够实时的刷新共享内存池，保证轧机二级通信子系统的正确性，在系统开发时，可以利用 ACE 的 Reactor 组件，通过反应器调度定时器的方法，对超时事件进行超时处理，如图 4.10 所示，从而实现对共享内存的定时刷新。

通过定义 ACE_Event_Handler 的子类 timer_handler 来处理来自 ACE_Reactor 的超时事件，使用这种实现方法，不仅可以方便的更改定时器的时间间隔，对定时器进行控制，而且在定时器超时时，不需要创建新的进程完成超时处理，而是采用直接调用的方法处理超时事件，节约了系统资源，另外由于采用的是 ACE_Reactor 组件，在可移植性方面

也有明显的优势。

```
#include <ace/Reactor.h>
#include <ace/Event_Handler.h>

class timer_handler : public ACE_Event_Handler
{
public:
    virtual int handle_timeout (const ACE_Time_Value &current_time,const void *act = 0);

    virtual int handle_close(ACE_HANDLE handle, ACE_Reactor_Mask close_mask)
    {
        cout << "handle_close" << endl;
        delete this;
        return 0;
    }

private:
        ~timer_handler()
        {
        }
};

ACE_Time_Value time_(4);
ACE_Reactor::instance()->schedule_timer(new timer_handler, NULL, time_, time_);

while (true)
{
            ACE_Reactor::instance()->handle_events();
}
```

图 4.10 基于 ACE_Reactor 的超时处理

Fig.4.10 The timeout handling based on ACE_Reactor

PrecalcStart 方法主要负责启动预计算，dispatch 在收到 papRcv 进程发送的上行报文数据后，会触发 precalc 进程进行预计算。dispatch 进程在预计算完成后会触发 papSnd 进程，将预计算的结果发送给接口进程，calcResultHMI 方法主要负责向人机界面发送预计算结果，使得操作人员可以根据得到预计算结果及时的做出调整。除了这两种方法外，dispatch 还会触发类似的许多方法，如 calcResultACC、calcResultPGL 等，这些方法的定义都在 dispatch 的 IDL 文件中，其定义方法与 papRcv 进程的 IDL 文件定义类似，只是其中包含的函数有所不同，这样以来通过 TAO 函数的调用可以实现进程间快速的通信，确保了轧机二级通信子系统的实时性。

4.2.3 发送进程的实现

发送进程 papSnd 是主要负责向接口进程发送下行报文数据，在轧机二级通信子系统的运行过程中，根据实际需要实时的将共享内存池或者数据库中的数据以报文的形式发送给接口进程，达到对基础自动化级设备的控制。papSnd 进程的交互图如图 4.11 所示。

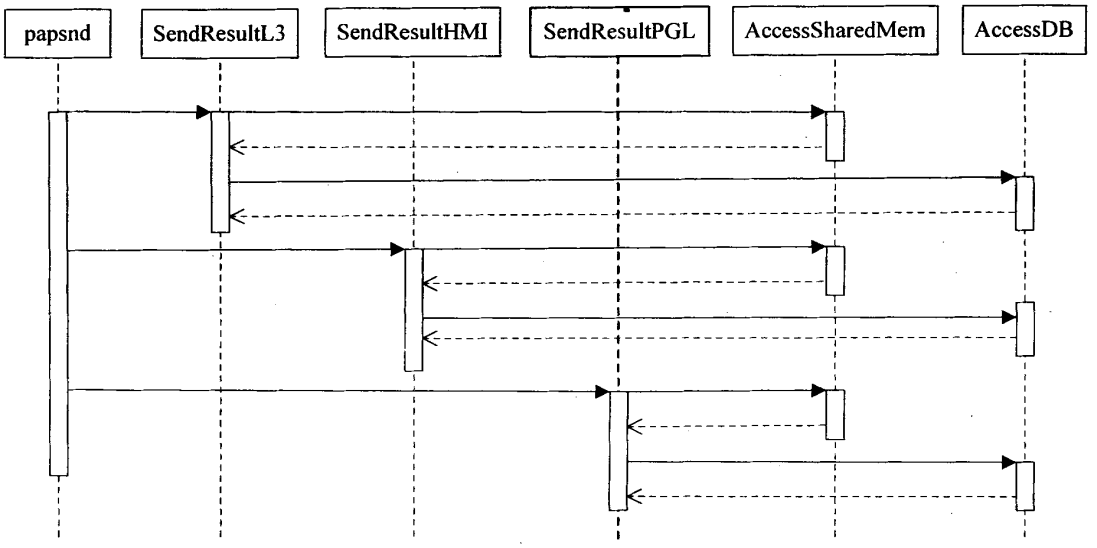

图 4.11 papSnd 进程的交互图
Fig.4.11 The interaction diagram of papSnd process

轧机二级通信过程中，dispatch 进程根据预先设计好的触发条件，触发 papSnd 进程访问共享内存池或者数据库，然后将获得的数据信息转化为报文发送到对应的接口进程，如 l3l、pgl 等，papSnd 访问共享内存的方式与 papRcv 进程的方式相似，都是利用 find 方法进行名字解析，papSnd 访问数据库同样采用的是 OO4O 数据库访问方式，这里不再赘述其具体实现过程。

4.2.4 预计算进程的实现

预计算进程 precalc 的功能是对尚未开始轧制的板坯进行完整的轧制规程计算，其数据源分别来自共享内存池和数据库中，数据库中存放的主要是板坯的原始数据和各种模型参数，而共享内存中的数据源常常包含当前板坯轧制状态的信息。在获取数据源后，precalc 进程将对板坯进行规程计算，计算完成后通知 dispatch 进程触发 spt 进程，向基础自动化级发送规程设定值，同时将计算后生成的日志文件保存与数据库。图 4.12 给出了预计算进程的交互图。

percale 进程主要有以下几个触发条件：

（1）出炉/入炉。在板坯出入加热炉时，dispatch 会收到来自加热炉的报文，然后触发 precalc 进程进行预计算，计算完成后再通知 dispatch 进程。

（2）人工操作请求。操作工根据板坯的实际轧制状况有时需要对轧制规程进行重新计算，通过人机接口发送预计算请求，使得 dispatch 进程触发 precalc 进程进行预计算。

（3）测温仪。在板坯经过测温仪时，根据测温仪的实测数据也可能根据实际需要

进行预计算，所以测温仪也是 precalc 进程的一个触发条件。

除了上述三个触发条件，还有一些因素会触发 precalc 进程进行预计算，根据实际的生产状况，在系统的开发中进行一定的增减。

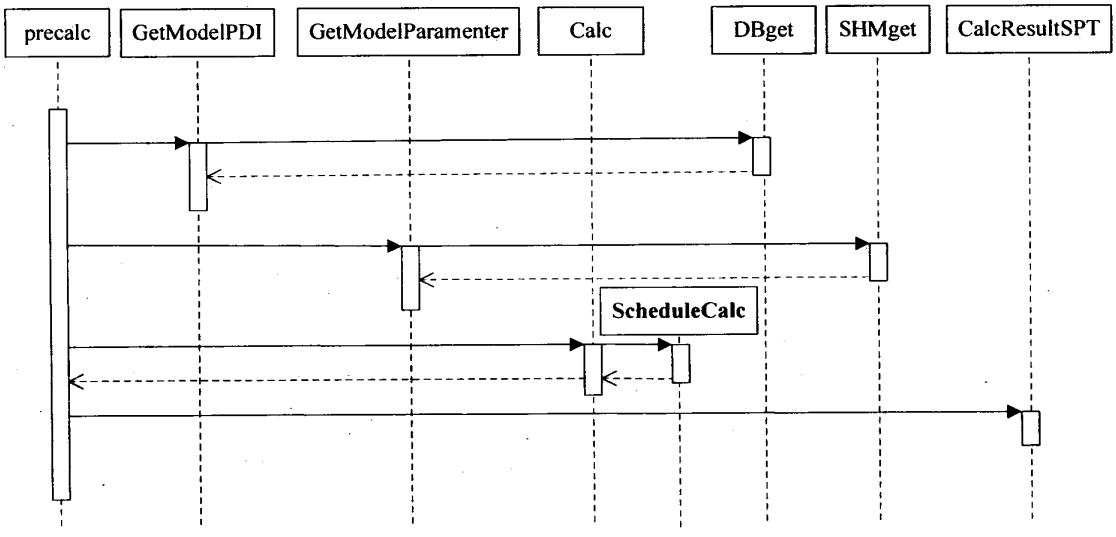

图 4.12 precalc 进程的交互图
Fig.4.12 The interaction diagram of precalc process

预计算进程访问共享内存池采用的方法和 papRcv 进程的访问方法相同，只是具体的函数名称和响应方式不同。precalc 进程也是通过 OO4O 数据库访问技术对 Oracle 10g 进行数据的查询与修改。预计算完成后，precalc 进程需要通过 spt 进程向基础自动化级发送规程设定值，在 precalc 的实现中只需调用相关的 TAO 函数，函数的具体实现由 dispatch 进程统一完成，其原理同 papRcv 进程一样，都是使用 ORB（对象请求代理）的通信方式。

由于前文对相关方法的实现进行了具体详细的说明，这里就不再赘述。

4.2.5 数据库的实现

本文提出的中厚板轧机二级通信平台调度监控系统在开发过程中的使用的数据库采用的是 Oracle 10g，为了提高对 Oracle 10g 的操作效率以及后期对整个数据库的维护管理，这里还还是用了 Database Configuration Assistant 和 PLSQL Developer，Database Configuration Assistant 主要负责对数据库的创建和维护。

PLSQL Developer 主要用于对数据库的操作，包括创建管理表空间、创建管理表，以及对表进行插入、修改、查询等操作，利用 PLSQL Developer 使得数据库的操作变得简单方便，更加直观的反应了数据库数据的结构信息。

本文提出的中厚板轧机二级通信平台调度监控系统中数据库部分的实现主要包括一下几个部分：

（1） 创建数据库 NEURollDB，用来对整个系统中的数据进行永久性存储。

（2） 创建表空间 ROLLS，用来存放数据库中的表，表空间是数据库的一个逻辑划分。

（3） 创建表 LOG、表 STEEL_COMPOSITION、表 FU_L2_CHARGE 等，表中的每个属性都是根据生产中的实际需要进行设计的。

为了明确的阐述数据库中各个表的具体含义与功能，这里主要对以下几个表进行了详细的介绍。

（1）图 4.13 给出了表 LOG 的定义，表 LOG 主要生成轧机二级通信子系统中的通信日志，记录系统中各个进程的消息数据，以便工作人员能够及时的对系统进行调试，同时也为日后的数据查询工作提供帮助。表 LOG 中的 TIME 属性使用的数据类型是 DATE，默认为系统时间。

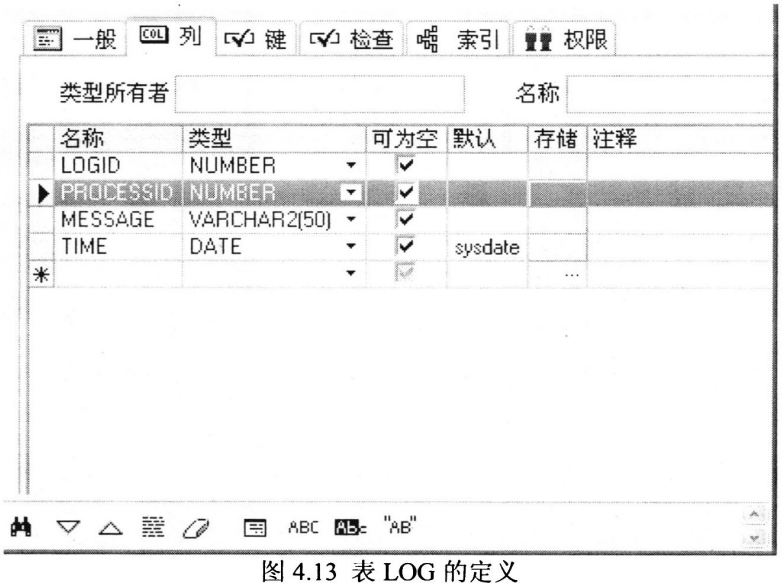

图 4.13 表 LOG 的定义

Fig.4.13 The definition of table LOG

（2）图 4.14 给出了表 PDI 的定义，表 PDI 主要存储了钢板的原始数据，包括板坯的长、宽、厚，目标钢板的长、宽、厚，出炉温度、开轧温度、终扎温度等。其中属性 PLATEID 是表 PDI 的主码，其他大部分属性的数据类型都为 NUMBER。

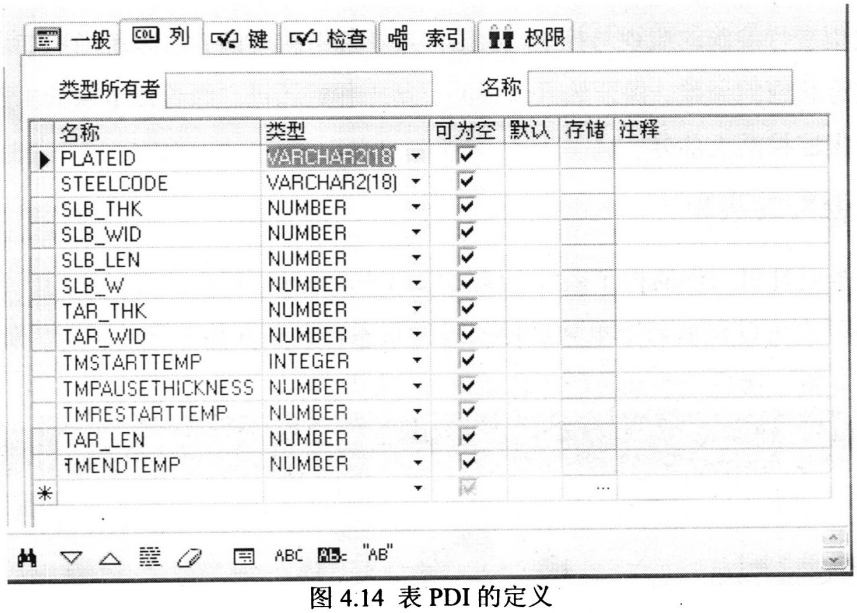

图 4.14 表 PDI 的定义

Fig.4.14 The definition of table PDI

（3）图 4.15 给出了表 HMI_L2_OPFINISHING 的定义，表 HMI_L2_OPFINISHING 主要反应了 HMI 对轧机二级的精轧控制参数，例如抛钢方向设定、冷却方向选定、以及最大下压量控制等。

图 4.15 表 HMI_L2_OPFINISHING 的定义

Fig.4.15 The definition of table HMI_L2_OPFINISHING

除了以上几个表之外，这个系统的数据库还定义了许多表，这些表都是根据实际生产需要设计的，各个表的功能也有所不同，在此不再一一列举。

4.3 轧机二级监控子系统的实现

轧机二级通信具有复杂性、实时性等特点，轧机二级监控子系统的目的就是要对系

统中各个进程进行准确的监视与控制，让工作人员能够实时的查看系统当前的运行状况并及时的进行相应的调整，保证轧机二级的正常工作。轧机二级监控子系统主要包括进程监控和数据监控两大部分。

4.3.1 进程监控的实现

进程监控是轧机二级监控子系统中重要的部分，通过进程监控可以控制所有进程的开启与关闭，也可以根据每个单独进程的运行状态，调整其他进程的运行。如图 4.16 所示，就是轧机二级通信子系统的进程监控部分的实现。

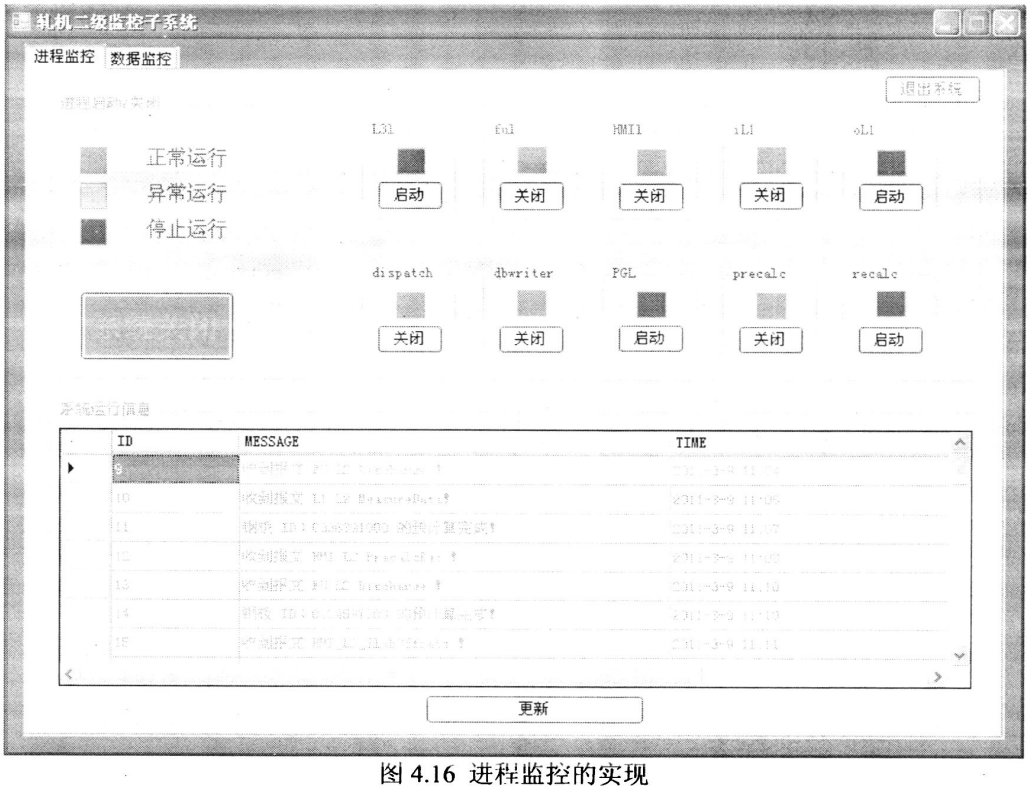

图 4.16 进程监控的实现

Fig.4.16 The implementation of process monitor

通过本文前面的介绍，轧机二级中的各个进程的启动是有一定的先后顺序的，例如 dispatch 进程是守候进程，必须最先启动，dbwriter 进程是访问数据库进程，在其他进程访问数据库前必须启动 dbwriter 进程等等。批处理文件，是一种文本文件，简单的说，它的作用就是能够自动的连续执行多条命令。轧机二级监控子系统中的进程监控部分，就是通过系统启动与关闭按钮的事件响应机制，调用批处理文件，启动或关闭所有进程，这样以来，不仅对进程的启动关闭顺序进行控制，也减少了逐一操作每个进程带来的麻烦，缩短系统启动与关闭的时间。

进程监控部分还可以对每个单独进程的运行状态进行实时的监视，操作人员可以根

据每个进程的运行状态，调整控制其他进程的启动与关闭。例如，如果 dispatch 进程的状态异常，则需要重新启动系统的所有进程，因为 dispatch 是守候进程负责轧机二级通信子系统中所有进程的调度以及创建维护共享内存池，它的异常状态会造成其他所有进程运行错误。再比如，如果 precalc 进程运行状态显示异常，则需要重新启动 ill 进程向共享内存或者数据库重新传递预计算参数，避免 precalc 进程“读脏”。Windows 窗体应用程序提供了大量的控件，这些控件配合一定的事件响应机制，可以有效的完成进程监控的任务。常见的事件响应包括按钮的事件响应、目录的事件响应等，在不同的事件响应函数中，通过对控件属性的改变显示的反应进程运行的状态信息，也可以通过函数的调用控制进程的开启与关闭。

```
private: System::Void button_HMIL_Click(System::Object^  sender, System::EventArgs^  e)
{
    if(flag_HMIL==0)
    {
        button_HMIL->Text="关闭";
        text_HMIL->BackColor=BackColor.Green;//改变背景颜色
       WinExec("../../../Debug/HMIL.exe", SW_SHOWMINIMIZED); //启动进程，激活窗口并将其最小化
        flag_HMIL=1;
    }

    else if(flag_HMIL==1)
    {
        button_HMIL->Text="启动";
        text_HMIL->BackColor=BackColor.Red;
       KillExe(L"HMIL.exe"); //关闭进程,调用KillExe函数，注意字符的转换
       flag_HMIL=0;
    }
}
```

图 4.17 hmil 进程的事件响应

Fig.4.17 The incident response of the hmil process

hmil 进程事件响应的实现如图 4.17 所示，button_HMIL_Click 函数是事件响应的入口，在函数体内部，通过对控件 button、控件 test 属性的改变，表示进程的运行状态，同时利用 WinExec 方法和 KillExe 方法分别对进程进行单独的启动与关闭。在轧机二级系统中包括许多进程，它们的事件响应方式与 hmil 进程类似，只是函数参数和控件操作的不同，这里不再一一介绍它们的实现方法。

进程监控中的系统运行信息，可以按时间顺序将系统中的通信状况呈现给操作人员，便于操作人员及时的了解系统内部的运行情况，另外这些消息信息还可以存放在数据库中，形成系统日志，便于日后的查阅。

4.3.2 数据监控的实现

数据监控主要是便于工作人员对数据库中的数据表进行快速查询，例如对板坯原始数据的查询、对模型预计算参数的查询以及对系统日志的查询等。出于安全性角度，数据监控部分只允许工作人员对数据进行查询操作，而不允许更新和修改，如图 4.18 所示，是轧机二级监控子系统数据监控部分的实现。

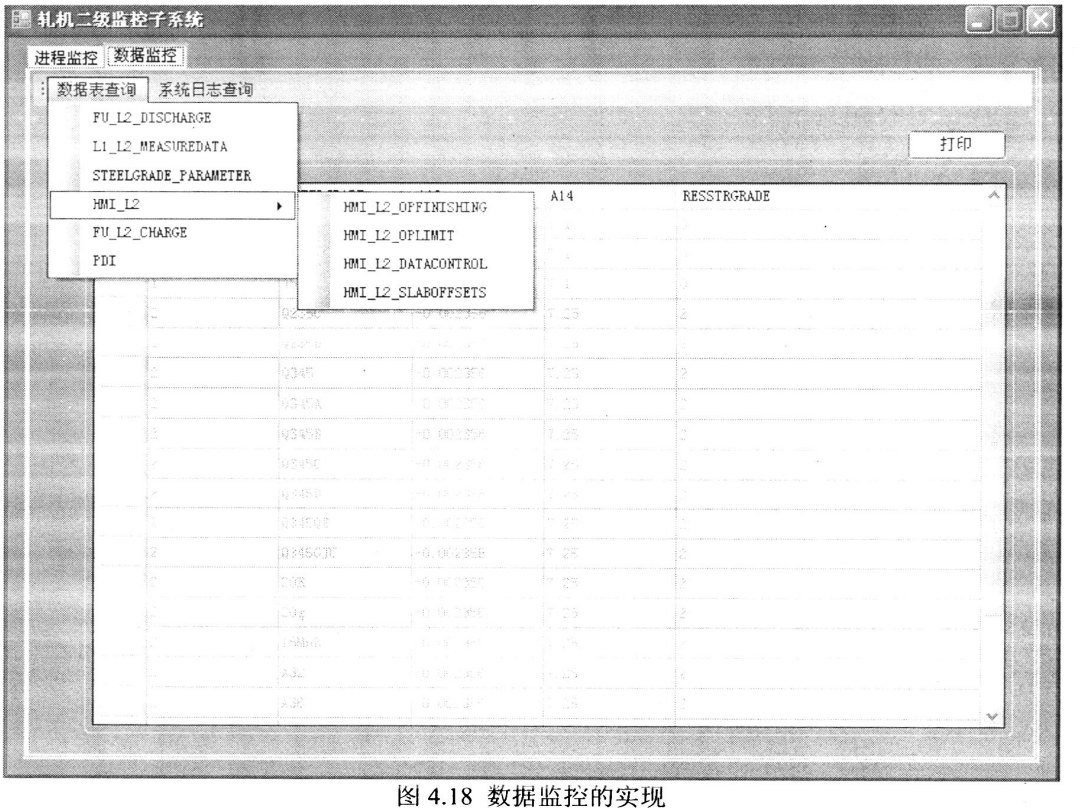

图 4.18 数据监控的实现

Fig.4.18 The implementation of data monitor

数据监控部分的实现主要使用了 Windows 窗体应用程序中的 DataGridView 控件，用来显示不同的数据表，在实现过程中，需要对 DataGridView 控件的 DataSource 属性进行设置，将它绑定到指定的数据源，然后再设置 DataMember 属性，从而绑定到每一个数据表。操作人员每次查询时，通过事件响应机制调用预先编写的的 dtshowdata 函数将 DataSource 绑定到相应的数据表，在通过 ODBC 连接指定的数据库实现数据的快速查询，另外通过对 DataGridView 控件的其他基本属性的设置，DataGridView 控件处于“只读”状态，防止操作人员对数据的修改，保证数据监控的正确性。

4.4 轧机二级自学习子系统的实现

轧机二自学习子系统主要通过对海量实测数据的分析，提高模型的计算精度，其具体功能包括：零点修正、轧制力长期自学习和钢板温度修正。本节主要介绍了轧制力长

期自学习的实现过程。

4.4.1 数据仓库的实现

数据仓库是面向主题的，主题是数据仓库中数据分类的标准，构建一个数据仓库必须从物理实现、数据抽取、转换、装载、数据更新等逐步进行。本文基于 OWB 11g 具体实现了轧机二级自学习子系统中数据仓库的构建。

（1）事实表

表 4.2 轧制力主题的事实表

Table 4.2 The fact table of roll force

序号	字段名	含义	类型	备注
1	TIME_ID	时间代码	NUMBER（8）	
2	PLATE_ID	钢板代码	NUMBER（10）	
3	TEMP_ID	温度代码	NUMBER（3）	
4	ROLL_FORCE	轧制力	NUMBER（8）	

（2）时间维

表 4.3 时间维表

Table 4.3 Time dimension table

序号	字段名	含义	类型	备注
1	TIME_ID	时间代码	NUMBER（8）	
2	YEAR_ID	年	NUMBER（4）	
3	QUANTER_ID	季度	NUMBER（2）	
4	MONTH_ID	月	NUMBER（2）	
5	DAY_ID	日	NUMBER（2）	

（3）温度维

表 4.4 温度维表

Table 4.4Temperature dimension table

序号	字段名	含义	类型	备注
1	TEMP_ID	温度代码	NUMBER（3）	
2	TEMP_CLASS	温度类别	NUMBER（10）	
3	TEMP_VALUE	温度值	NUMBER（10）	

（4）钢板维

表 4.5 钢板维表

Table 4.5 Plate dimension table

序号	字段名	含义	类型	备注
1	PLATE_ID	钢板代码	NUMBER（10）	
2	PLATE_NAME	钢板名称	VARCHAR2(16)	

表 4.2 到表 4.5 给出了数据仓库中以轧制力为主题的事实表与维度表，对数据仓库进行了物理实现。

数据仓库的物理实现后需要对数据源进行抽取、转换和装载，在轧机二级生产的海量数据中，按照数据仓库的物理设计抽取相关的数据，并按照一定的转换方法对数据进行转换，如表 4.6 所示，给出了数据转换的具体规则，数据经过一系列的处理后最终装载到数据仓库中。使用 OWB 11g 构建 ETL 可以从多个数据源提取数据，降低了数据仓库开发的复杂性，另外 OWB 11g 还具备标准开放的结成框架和接口，便于与其他软件的集成使用。

表 4.6 数据转换规则

Table 4.6 The rules of data conversion

需要转换的数据对象	转换规则
时间字段	统一转换成“年-月-日”的表示形式
备用字段、冗余字段	合并成一个字段，压缩空间，消除不一致性
轧制力字段	统一保留到小数点后两位

在配置好 OWB 11g 以后，就可以利用 OWB 11g 建立事实表与维度表的关联，图 4.19 给出了 OWB 中已经建立好的维、立方和映射。

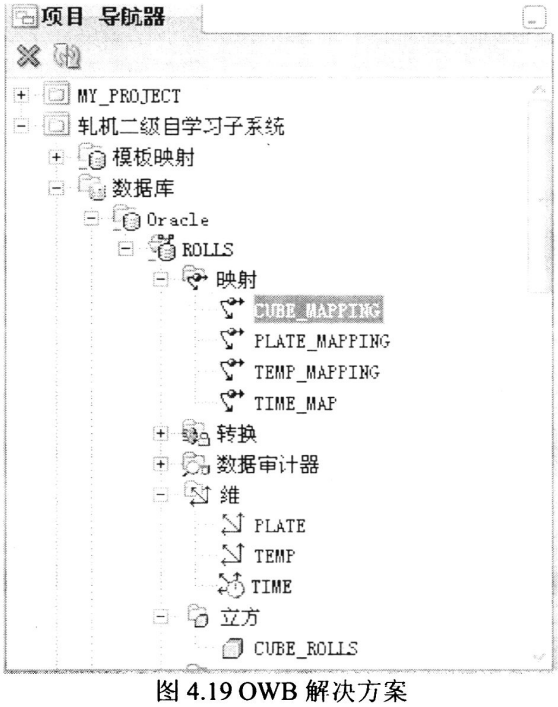

图 4.19 OWB 解决方案

Fig.4.19 The OWB solutions

通过 OWB 中的 Mapping，可以将数据源中的数据映射到各个维度表，在映射完成后可以通过“验证”来检查已经定义的映射是否存在错误，如果存在错误，系统会给出验证信息，提示错误发生的位置和具体错误类型，通过“生成”还可以产生实际操作过程中产生的代码，最后经过“部署”完成最后的 ETL。如图 4.20 所示，是通过 OWB 进程流完成了最终的数据立方体的构建。

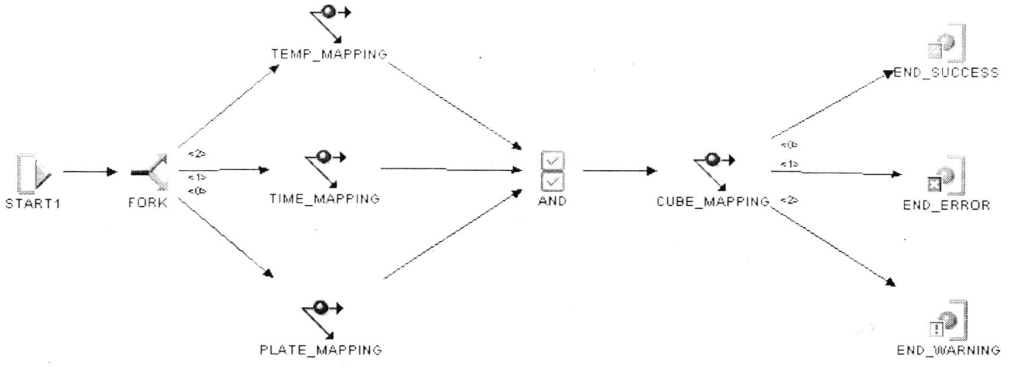

图 4.20 创建数据立方体进程流的实现
Fig.4.20 The implementation of workflow for creating data cube

4.4.2 OLAP 的实现

轧机二级自学习子系统的 OLAP 模块利用了 Oracle 11g 的 OLAP Java 接口对数据仓库进行了多维数据分析，实现 OLAP 的功能。

OLAP 分析处理模块与数据仓库的连接主要通过 JDBC 的数据库访问方式，图 4.21 给出了方法实现的关键部分。

```
Class.forName("oracle.jdbc.driver.OracleDriver");
String URL = "jdbc:oracle:thin:@172.16.168.161：1521：renxiaodong";
String user = "ROLL";
String password = "ROLL";
Oracle.jdbc.OracleConnection conn = (oracle.jdb.OracleConnection)
Java.Sql.DriverManager.getConnection (URL, user, password);
ExpressTransactionProvider TanP = new ExpressTransactionProvider ( );
ExpressDataProvider DatP = new ExpressDataProvider (conn, TanP);
DatP.initialize ( );
```

图 4.21 JDBC 访问数据库
Fig.4.21 Access database by JDBC

数据仓库中的元数据包括维度、层次、属性等数据结构，是一个数据机构的集合，OLAP 可以对元数据进行访问，从而提高数据仓库中数据的利用率，具体实现方法如图

4.22 所示。

```
MdmMetadataProvider DWP = null;
DWP = (MdmMetadataProvider)dp.getDefaultMetadataProvider( );
MdmSchema root = DWP.getRootSchema( );
List sims = schema.getDimensions( );
List DubSchemas = schema.getSubSchemas( );
```

图 4.22 OLAP 对元数据的访问
Fig.4.22 Access metadata for OLAP

联机分析处理 OLAP 的数据查询过程需要使用 Source 类进行实现，Source 类提供了许多方法，如：求和、求最大、最小值、升序、降序等，使用这些方法对多维数据模型进行分析，多角度多方位的考察数据的潜在信息。具体的实现方法如图 4.23 所示。

```
Source timeSource = timeDim.getSource ( ) ;
//通过Source的方法来创建对象, join方法来实现查询
Source newSource = forceinfo.join(timesDim).join(tempDim);
//表示从时间维和温度维角度查询轧制力信息。
Source timesforce= timesDim.selectValue("2010");
//表示只选择2010年的数据
```

图 4.23 OLAP 的数据查询方法
Fig.4.23 The query methods for OLAP

4.5 系统环境配置

在对中厚板轧机二级通信平台调度监控系统进行开发前，需要对系统需要的开发工具进行正确的安装，例如 ACE、TAO、Oracle 10g 数据库、PLSQL Developer、ActivePerl 等，在正确的安装后还要对系统环境进行进一步的配置才可使用，本章主要介绍了在 Visual Studio 2008 下进行 ACE+TAO 的环境配置以及 OWB 11g 的环境搭建过程。

4.5.1 ACE+TAO 环境配置

为了配合 TAO 的使用，ACE 的版本不能高于 TAO 的版本，这里使用的是相同版本的 ACE+TAO 组合，具体配置过程如下：

（1）解压 ACE+TAO 的压缩文件到磁盘根目录，如 d :\ACE_wrappers，设置系统环境变量 ACE_ROOT=D:\ACE_wrappers 和 TAO_ROOT=%ACE_ROOT%\TAO，同时在 PATH 变量中添加%ACE_ROOT%\bin;和%ACE_ROOT%\lib。

（2）在 d :\ACE_wrappers \ace 目录下新建一个空文件 config.h，里面包含内容为：#include "ace/config-win32.h"，表明要在 win32 平台下进行编译。

（3）打开 Visual Studio 2008，在 Tools->Options->Projects and Solutions

->VC++Directories 中加入以下内容：

Executable files 中添加：

$(ACE_ROOT)\bin，

Include files 中添加：

$(ACE_ROOT)，$(TAO_ROOT)，$(TAO_ROOT)\orbsvcs

Library files 中添加：

$(ACE_ROOT)\ace，$(TAO_ROOT)\tao，$(TAO_ROOT)\orbsvcs\orbsvcs

（4）执行%TAO_ROOT%\TAO_ACE_vc9.sln，重新编译这个 solution，经过漫长的等待，就可以得到开发过程中需要的 ACE 和 TAO 相关的链接库等文件。

4.5.2 OWB 环境搭建

在完成了 Oracle 10g 和 OWB 11g 的安装后，需要对 OWB 11g 的环境进行搭建才能完成对数据仓库的使用，具体步骤如下：

（1）创建一个名为 orcl 的数据库。本文使用的 SYSDBA 用户名/口令为 sys/oracle。

（2）通过“开始>运行>cmd.exe”以 SYSDBA 权限的 SYS 用户身份调用 SQL Plus，例如，您可以在命令行键入：sqlplus sys/<sys 口令>as sysdba 发出以下命令创建 OWBSYS：@<OWB 主目录>/owb/UnifiedRepos/cat_owb.sql;按 Enter 键。系统将提示您为 OWBSYS 用户指定表空间，键入 users 并 Enter 键。

（3）以 SYSDBA 权限的 SYS 用户身份调用 SQL Plus，在命令提示符下键入以下命令：SQL>@c:\oracle\OWB_home_11g\owb\UnifiedRepos\reset_owbcc_home;按 Enter 键。系统将会提示您输入安装控制中心 ControlCenter 的 Oracle 主目录的完整路径，输入 C:/oracle/OWB_home_11g 按 Enter 键。

（4）继续执行 SQL Plus 命令，解除 OWBSYS 帐户的锁定并指定一个名为 OWBSYS 的口令：alter user OWBSYS account unlock;alter user OWBSYS identified by OWBSYS;

（5）使用 Repository Assistant 创建信息库，在 Database Information 窗口中，输入 localhost 作为主机名，输入 1521 作为端口号，输入 orcl 作为 Oracle 服务名，单击 Next；在 Choose Operation 窗口中，选择 Manage Warehouse Builder workspaces 选项，单击 Next；在 Choose Workspace Operations 窗口中，选择 Create a new Warehouse Builder workspace，单击 Next；在 New or Existing User 窗口中，单击 Create a workspace with a new user as workspace owner，单击 Next；在 DBA Information 窗口中，输入 system 作为用户名，输入 oracle 作为口令，单击 Next；在 Workspace Owner (New)窗口中，输入 owb/owb 作为工作区所有者用户名/口令，确认口令为 owb，输入 my_workspace 作为工作区名，单

击 Next；在 OWBSYS Information 窗口中，输入 owbsys/owbsys 作为用户名和口令，单击 Next；在 Select Tablespaces 窗口中，您可以指定要用于数据、索引、临时表和快照的表空间，在这里，您使用默认的表空间，单击 Next；Workspace Users 窗口允许你选择一个已经存在的数据库用户并将其注册为一个 Warehouse Builder 用户。该用户具有在 Design Center 中工作的权限，并且能够通过 Control Center 部署和执行对象，另外，也可以创建一个新的数据库用户。

（6）经过前面几个步骤，最后进入总结窗口，查看信息，点击完成。至此 OWB 11g 的环境搭建基本完成，进入登陆界面。

根据实际开发情况的需要，为了简化安装，在环境搭建过程中将 Oracle 10g 和 Oracle Warehouse Builder 11g 安装在同一台计算机上，需要注意的是，在安装过程中要将 Oracle 10g 数据库和 OWB 11g 安装到不同的根目录下，防止安装出错。

4.6 本章小结

本章根据前文提出的系统设计方案，结合实际的生产条件，基于 ACE、TAO、OO4O、OLAP 等关键技术对中厚板轧机二级通信平台调度监控系统进行了开发，同时介绍了 OWB、ACE+TAO 等关键技术的环境配置方法，通过对通信子系统、监控子系统、自学习子系统逐个实现，最后完成这个轧机二级通信调度监控系统的构建。

第5章 测 试

本章对本文完成的中厚板轧机二级通信平台调度监控系统进行测试，其中包括通信功能测试、监控功能测试以及自学习功能测试三个部分，从而验证本文提出的基于 ACE、TAO、OLAP、OO4O 等关键技术的轧机二级通信调度监控系统设计方案的正确性和有效性。

5.1 测试环境

由于本课题得到东北大学轧制技术及连轧自动化国家重点实验室开放课题《中厚板轧机二级系统开发》的帮助，所以将硬件环境选为东北大学轧制技术及连轧自动化国家重点实验室西门子轧机 PLC 测试平台，实现轧机一级与轧机二级通信接口的互联，提高了测试结果的准确性。另外将本文实现的中厚板轧机二级通信平台调度监控系统部署在小型服务器（CPU：四核，3.20GHz，内存：4GB，硬盘：500G，系统：Windows Xp）上，完成最终的测试环境的搭建。

5.2 通信功能测试

接口进程主要负责在工业以太网中实现轧机二级与轧机一级、轧机三级服务器进行通信，主要的接口进程包括：pgl、accl、L3l 等，本次测试的数据都是由从东北大学轧制技术及连轧自动化国家重点实验室西门子轧机 PLC 平台发送来的，主要采用的是 TCP/IP 协议进行数据传输。

图 5.1 ful 进程的通信功能测试

Fig.5.1 The test of communication function of ful process

如图 5.1 所示，是对接口进程 ful 的通信功能测试，在没有收到 PLC 发送数据或者

没有数据要通过 ful 接口向外部服务器发送时，ful 进程处于等待状态。

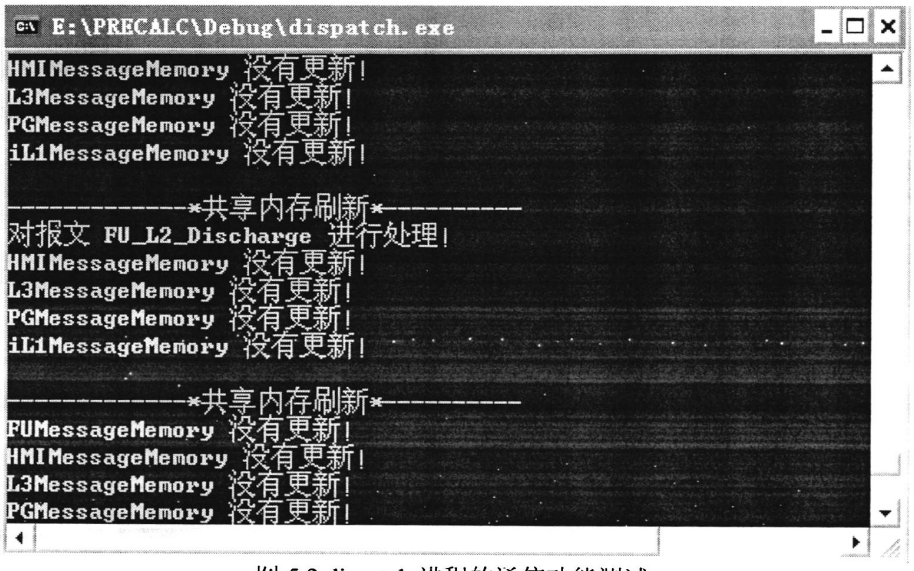

图 5.2 dispatch 进程的通信功能测试

Fig.5.2 The test of communication function of dispatch process

当收到来自 PLC 的数据时，ful 进程会触发 dispatch 对共享内存池中的 FUMessageMemory 共享内存块进行更新，如图 5.2 所示。

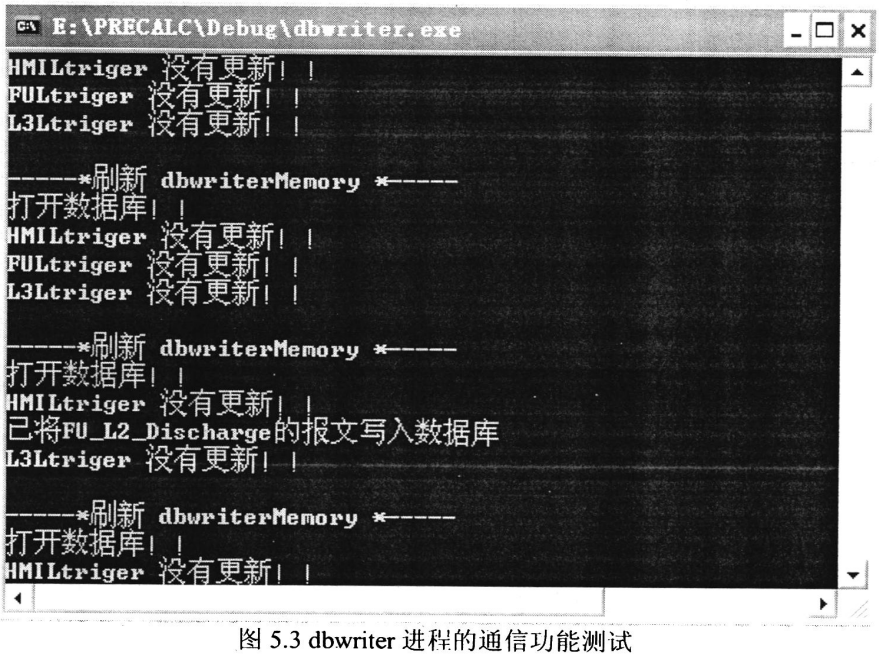

图 5.3 dbwriter 进程的通信功能测试

Fig.5.3 The test of communication function of dbwriter process

dispatch 进程还会调度 dbwriter 进程，将来自 ful 进程的数据写入数据库，进行消息备份，如图 5.3 所示。

以上是对接口进程 ful 通信功能的测试过程，包括 ful 接收来自 PLC 的上行报文，并触发 dispatch 进程对共享内存的更新以及对其他进程（如 dbwriter、precalc 等）的调

度管理等。其他进程的功能测试与此类似，例如 precalc 进程在预计算结束后触发 dispatch 进程调度 spt 进程将预计算结果发送给接口进程等，这里对其他进程的测试过程不再一一赘述。

5.3 监控功能测试

轧机二级监控子系统是整个中厚板轧机二级通信平台调度监控系统的重要组成部分，通过监控子系统，工作人员可以了解系统内部进程的通信关系和通信状态，以便根据实际需要进行及时的调整，使整个系统高效的运行。

本节对轧机二级通信子系统中进程监控部分的“系统运行信息”模块进行了现场测试，“系统运行信息”能够实时的反应进程间的通信状况，体现系统的运行状态，是轧机二级监控子系统中一个重要的部分。如图 5.4 所示，是对“系统运行信息”一段时间内的监控测试结果。

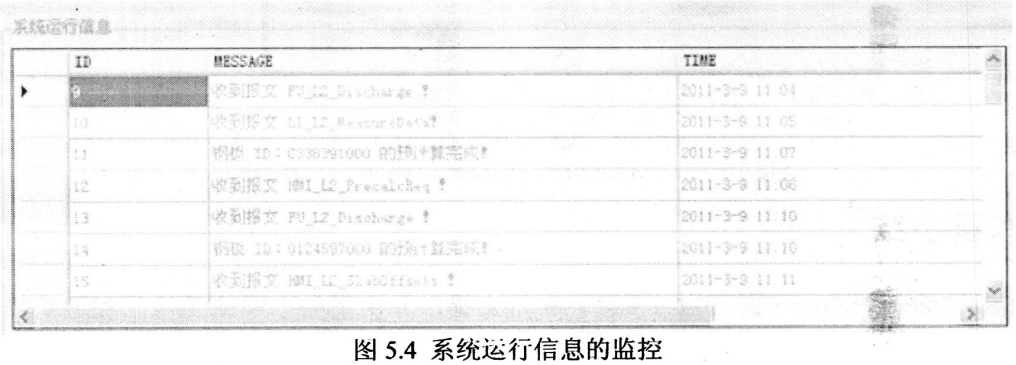

图 5.4 系统运行信息的监控

Fig.5.4 Monitoring the running information of the system

5.4 自学习功能测试

图 5.5 给出了数据仓库的性能测试结果，主要展示了有关轧制力的相关信息，根据数据仓库历史性、稳定性的特点，加上 OLAP 等分析工具的配合使用，可以对海量数据进行查询分析，形成轧制力的长期自学习，为提高模型计算的准确性带来很大帮助。

图 5.5 数据仓库性能测试

Fig.5.5 The test for performances of the data warhouse

5.5 本章小结

本章主要利用西门子的 PLC 平台，对本文提出并实现的中厚板轧机二级通信平台调度监控系统进行了测试，包括通信功能的测试、监控功能的测试以及自学习功能的测试三个部分，通过测试对测试结果的分析，进一步验证了本文提出的设计方案的正确性和可行性。

第6章 总结与展望

本章对全文内容进行了归纳，从整个中厚板轧机二级通信平台调度监控系统的需求分析、概要设计、详细设计到最后系统的实现与测试，总结了工作中遇到的问题和解决方法，并对后续工作进行展望。

6.1 总结

中厚板是基础设施建设和大型工程、重型机械制造的重要原材料，因其广泛的用途，在钢铁行业处于重要地位。随着轧钢自动化进程的加快，对轧机二级通信调度监控系统的需求日益增加，构建一个拥有自主知识产权的、跨平台、通用的、可扩展的中厚板轧机二级通信平台调度监控系统，具有深远的市场意义和社会意义。本文以国内某钢铁公司的中厚板轧机二级控制系统为研究背景，结合 ACE、TAO、OLAP、OO4O 等关键技术实现了中厚板轧机二级通信平台调度监控系统的开发。

本文对中间件、面向对象中间件、ACE、TAO、数据仓库等技术进行了深入的研究，并通过对国内某钢厂中厚板生产线的实地考察，对中厚板轧机二级通信调度监控系统进行了需求分析，在需求分析的基础上根据现有技术的掌握提出了系统的概要设计方案，包括系统框架的设计、数据通信方式的设计、数据流分析和数据库访问方式的讨论，然后根据实际功能需求分别对系统中的轧机二级通信子系统、轧机二级监控子系统和轧机二级自学习子系统进行了详细设计，最后在 windows xp 操作系统下，利用 Visual Studio 2008、Oracle 10g、OWB_11.2.0.1_Windows、PLSQL Developer 等开发工具实现了中厚板轧机二级通信平台调度监控系统的开发。系统实现后，在东北大学轧制技术及连轧自动化国家重点实验室的帮助下，结合东北大学轧制技术及连轧自动化国家重点实验室西门子轧机 PLC 测试平台，对系统进行了功能测试，包括通信功能测试、监控功能测试和自学习功能测试，验证了本文提出的中厚板轧机二级通信平台调度监控系统的正确性与可行性。

本文主要解决了一下关键问题：

（1）对轧机二级通信调度监控系统进行了合理的数据建模

轧机二级生产线在实际生产中是十分复杂的，轧机二级系统具有实时性的要求和分布式部署的特点，本文根据国内某钢铁企业的实际生产线的工作流程，提出利用 ACE、TAO、OLAP、OO4O 等关键技术的系统框架，使模型具有较高的通用性和可扩展性，从而满足实际生产的需要。

（2）实现了系统的跨平台的特性

ACE 可以使得开发人员快速的开发可移植性、高性能的应用程序，并包括了一些经典的设计模式。TAO 是一种在自适应通信环境 ACE 基础上的高性能实时 CORBA 中间件框架，在面向对象系统中提供了跨硬件平台、跨操作系统、跨编程语言和跨网络网络协议等远程调用功能。本文利用 ACE+TAO 解决了系统的可移植性问题，这也是轧机自动化生产研究领域的一个长期热门课题。

（3）解决了多进程通信的正确性

轧机二级通信子系统中，采用的是多进程通信的方式，各个进程间存在一定的依赖关系。本文提出了多进程控制序列，守候进程依照控制序列对各个进程实行调度与控制，保证多进程通信在通信时序上的正确性。

（4）数据的永久性存储和对数据库的快速访问

轧机二级系统中的部分数据需要永久性的存储，便于以后对数据信息的查询，Oracle 10g 是目前世界上最强的、最流行的数据库系统，它不但可以支持多用户的大事务量事务处理，同时也支持分布式的数据处理，具有很强的移植性，本文利用 Oracle 10g 数据库对轧机二级生产过程中的数据信息进行永久性存储，同时利用 OO4O 关键技术实现对数据库的快速访问，提高数据库的访问效率，最终满足轧机二级数据的永久性存储和快速访问的要求。

（5）模型的自学习

模型的自学习包括轧制力长期自学习、钢板温度自学习等，通过模型的自学习对现有模型进行修正，提高模型的计算精度。数据仓库不是简单的对大量数据进行存储，而是将来自不同数据源的数据进行抽取、转换、装载，形成一个长期的、稳定的数据信息集合；OLAP 联机分析处理，是对大规模数据进行复杂的分析，它为分析人员提供对数据的多角度观察和一致性交互操作，使得分析人员可以获得对数据信息的深入理解。本文提出的基于 DW+OLAP 轧机二级自学习子系统，具有强大的数据处理和分析能力，对生产过程中轧机二级的数据信息进行深入的分析，实现模型的自学习功能。

6.2 展望

本文提出的中厚板轧机二级通信平台调度监控系统，虽然基本满足了轧机二级平台的通信要求、监控调度要求等，但是在具体细节方面还存在一定的问题，需要进一步的研究和解决，比如数据精度问题、数据库海量数据的查询问题、监控子系统监控功能不足问题、数据分析能力不高问题等等，由于课题研究时间有限以及中厚板轧机二级实际生产的复杂性，本文实现的中厚板轧机二级通信平台调度监控系统只能够满足中厚板生

产线上轧机二级的主要功能，对于一些细节问题还需要进一步的完善，最终在现有系统的基础上不断的进行功能扩展和性能提高。以上提到的问题都是在今后对系统进行完善时必须认真研究和解决的实际问题。

参考文献

1. 王国栋. 中厚板轧制技术与装备[M]，北京：冶金工业出版社，2009，2-7.
2. 王国栋, 吴迪, 刘振宇等. 中国轧钢技术的发展现状和展望[J]，中国冶金，2009，19(12)：1-13.
3. 陈东. 基于 ACE 的中厚板轧机二级通信系统的设计与实现[D]，沈阳东北大学，2010.
4. Dong Chen, Guiran Chang. Design and Implementation of a Portable ACE-based IPC Platform[J], IEEE Computer Society, 2010, 121: 502-505.
5. 牛文勇, 李建平, 王君等. 首钢 3500mm 中厚板轧机 AGC 基础自动化系统[J]，冶金自动化，2006，1：25-37.
6. 矫志杰, 胡贤磊等. 中厚板轧机设定计算功能的在线实施[J]，东北大学学报(自然科学版)，2005，26(7)：644-647.
7. 李公法, 孔建益, 刘安中等. 基于 CORBA 的轧机远程监测与故障诊断系统[J]，机床与液压，2005，7：183-186.
8. Andreas Kugi, Kurt Schlacher, Rainer Novak. Onlinear control in rolling mills: A new perspective [J], IEEE Transactions on Industry Application, 2001,37, (5): 1394-1402.
9. Freyer B H, Craig I K, Pistorius P C. Gauge and tension control during the acceleration phase of a steckel hot rolling mill [J],ISIJ International, 2003,43 (10): 1562-1571.
10. Eugenio Brusa, Luca Lemma. Numerical and cxperimental analysis of the dynamic effects in compact cluster mills for cold rolling [J],Journal of Materials Processing Technology, 2009,209 (5): 2436-2445.
11. 李纪云, 董小社等. 分布式对象中间件技术[J]，计算机工程与设计，2004，2：170-173.
12. W. Emmerich, M. Aoyama, J. Sventek. The impact of research on the development of middleware technology [J], ACM Transactions on Software Engineering and Methodology, 2008, 17(4): 19(1)-19(48).
13. M. Nickschas, U. Brinkschulte. CARISMA-A Service-Oriented, Real-Time Organic Middleware Architecture [J]. Journal of Software, Sep 2009, 14(7): 654-663.
14. 朱其亮, 郑斌著. CORBA 原理及应用[M]，北京:北京邮电大学出版社，2001，30-119.
15. 卢立男, 周长春等. 一种集成 CORBA 与 Web Services 的中间件[J]，计算机系统应用，2011，4：131-135.
16. H. X. Wei, X. M. Duan, F. Y. Chen and X. L. Zhang. Research of open CNC system

based on CORBA [A].In Proc. of the 5th IEEE International Symposium on Embedded Computing [C], Piscataway: IEEE Computer Society, 2008, pages 364-369.

17. S. D. Huston, J. C. Johnson, U. Syyid 著，马维达译. AEC 程序员指南-网络与系统编程的使用设计模式[M]，北京：中国电力出版社，2004，65-119.
18. D. C. Schmidt. ACE: an Object-Oriented Framework for Developing Distributed Applications [A].In Proc. of the 6th USENIX C++ Technical Conference [C], Cambridge, Massachusetts, USENIX Association, April 1994, pages 1-17.
19. Douglas C. Schmidt, Stephen D.Huston. C++ Network Programming, Volumel[M]: Mastering Complexity with ACE and Patterns. Addison Wesley. 2002, 2-13,22-36.
20. Douglas C. Schmidt, Stephen D.Huston 著，马维达译. C++ 网络编程：基于 ACE 和框架的系统化复用（卷 1）[M]，电子工业出版社，2005，1：47-98.
21. 邓淳瑜．基于 ACE 架构的多终端远程接入服务系统研究[D]，大连大连海事大学，2010.
22. Object Computing, Inc. TAO Developer's Guide(Vol.1)[M], 2002,pages 2-33.
23. Object Computing, Inc. TAO Developer's Guide(Vol.2)[M], 2002,pages 49-147.
24. E. H. Binugroho, T. K. Ha, Y. B. Seo, J. W. Choi. Communication Architecture for AUV Test-Bed Using ACE/TAO Real-Time Event Channel [A]. In Proc. of the 27th Chinese Control Conference [C] , Piscataway: IEEE Computer Society, 2008, pages 260-264.
25. 蒋建平，何新华. 基于 Oracle 数据库 OO4O 技术的应用研究[J]，计算机工程与设计，2004，11：1996-1998.
26. 李新慧，杨培章等. OO4O 简介以及其在 VC++中的应用[J]，电脑编程技巧与维护，2005，5：34-42.
27. 柳莺，赵艳红等. 数据仓库技术研究和应用讨论[J]，计算机应用，2001，2：46-48.
28. Kimball R, Reeves L, Roos M, Thornthaite. The Datawarehouse Lifecycle Toolkit: Expert Methods for Designing. Developing and Deploying Datawarehouses, John Wiley&Sons; Inc. 1998.
29. 杨光，张雷. 数据仓库及联机分析处理技术[J]，计算机工程与科学，2000，1：39-42.
30. 王艳，张景峤. OLAP 技术在执法统计分析系统中的应用研究[J]，2011，7：1500-1502.
31. Douglas C. Schmidt, Stephen D.Huston. C++网络编程（卷 1）：运用 ACE 和模式消除复杂性[M]，武汉：华中科技大学出版社，2003，17-46.
32. Jun Ho Park, Myung Jin Lee, Soon Ju Kang. CORBA-based distributed and replicated

resource repository architecture for hierarchically configurable home network[J], Journal of Systems Architecture, 2005,（51）: 125-142.

33. 王名茗, 王卫平. 一种基于 TAO 和 ACE 的企业应用集成架构[J], 计算机工程, 2005, 31(16): 92-94.
34. Holmes, David, Maxwel. Data warehouse integration using best fit matching. Proceedings of the International Conference on Information and Knowledge Engineering, 2007, 17: 171-181.
35. 汤姆森著. 朱建秋等译. OLAP 解决方案：创建多维信息系统[M]，电子工业出版社，2004，9：3-87.
36. M. Kashyian. Portable Inter process Communication Programming [A]. In Proc. of the Second International Conference on Advanced Engineering Computing and Applications in Sciences [C], Piscataway: IEEE Computer Society, 2008, pages 181-186.
37. 李小群, 赵慧斌, 孙玉芳. 进程间通信机制的分析与比较[J], 计算机科学, 2002, (11): 16-21.
38. 蓝炳雄, 张丽. 基于 ACE 的共享内存的开发与研究[J]，计算机系统应用，2005，4：40-43.
39. 代流刚, 周昌玉. 基于网络通信的数据库访问技术在 ATS 中的应用研究[J]，计算机测量与控制，2011，2：436-438.
40. 聂文燕. 开发式数据库互连（ODBC）技术的探讨[J]，新疆职业大学学报，2004，12（1）：71-74.
41. 马奎, 李宏伟, 李勤超, 毛彪. .NET 下基于 OO4O 核心的 Oracle Spatial 空间数据互操作[J]，计算机应用，2009，29：205-208.
42. 张朝明等编著. 21 天学通 Oracle[M]，电子工业出版社，2010，5，27-59.
43. 张宁, 贾自艳等. 数据仓库中 ETL 技术的研究[J]，计算机工程与应用，2002，24：213-216.
44. C. Chen, X. Yan, F. Zhu, J. Han, and P. S. Yu. Graph OLAP: Towards online analytical processing on graphs. In ICDM, 2008, pages 103-112.
45. OWB[EB/OL], http://baike.baidu.com/view/1726009.htm, 2008, 7.
46. 李涛, 李慧, 谷建华, 潘慧芳. 基于 ACE 的并发编程模式和池式内存分配的研究[J]，计算机工程与设计，2006，27(1)：26-28.

致谢

首先，我要对我的指导老师常桂然教授表示衷心的感谢，本论文的撰写和研究工作都是是在他的悉心指导下完成的。攻读硕士学位二年以来，常老师对我的学习和科研给予了极大鼓励，在生活中给了我亲切的关怀和帮助，为我的成长倾注了大量的心血。常老师为人朴素认真、实事求是的工作态度深深的影响了我，使我在科研、学习、为人等方面均有了很大提高。常老师开阔的视野、严谨的治学态度和谦和的人格魅力给我留下了极为深刻的印象，使我领略到真正的学者之风，我在硕士阶段取得的成绩都凝聚这常老师的指导与关怀。无论是以后继续学习还是踏上工作岗位，他的谆谆教导我都会会铭记于心，受用终生。

其次我要感谢我的学长陈东博士，对于本课题的实验和本论文的撰写他都给予里极大的支持，帮助我解决了许多论文撰写过程中的难点，他做事认真、对待学术一丝不苟的精神是我学习的榜样。

我还要感谢东北大学轧制技术及连轧自动化国家重点实验室对本课题的支持。同时更要感谢该实验室的王国栋院士、崔海涛博士等对本课题研究过程中所提供的支持和帮助。同时，我要把真诚的感谢送给我的父母和亲戚朋友，以及那些支持我的同学们，感谢他们在我的就学生涯中给予的鼓励和帮助。

感谢信息科学与工程学院各位老师的辛勤授课和指导。

最后，再次感谢所有给与我帮助的人，祝愿我的父母身体健康，祝愿老师们桃李满天下。

攻读学位期间发表的论著和科研、获奖情况

硕士期间参加的科研项目：

东北大学轧制技术及连轧自动化国家重点实验室开放课题《中厚板轧机二级平台开发》，2010.5-2011.4.

硕士期间发表的论文：

Dong Chen, Lizhong Jin, Xiaodong Ren, Guiran Chang, Fengyun Li. A Novel Secure Framework for Internet of Things._The Fifth International Conference on Genetic and Evolutionary Computing. (Ei, accepted)

硕士期间的获奖情况;

1. 2009-2010 东北大学研究生奖学金
2. 2010-2011 东北大学研究生奖学金