**内部**

中国科学技术大学

工程硕士学位论文



微软小娜性能分析系统

的设计与实现

作者姓名： 吕 阳

学科专业： 软件工程

校内导师： 郑启龙 副教授

企业导师： 杜奕瑾 高级工程师

完成时间： 二○一六年八月二日

**Limited**

University of Science and Technology of China

A dissertation for master’s degree

of engineering



Design and Implementation of Performance Analysis System for Cortana

Author’s Name： Yang Lv

Speciality： Software Engineering

Supervisor： Assoc. Prof. Qilong Zheng

Advisor： Principal Development Lead Ethan Tu

Finished time: August 2nd, 2016

中国科学技术大学学位论文原创性声明

本人声明所呈交的学位论文,是本人在导师指导下进行研究工作所取得的成果。除已特别加以标注和致谢的地方外，论文中不包含任何他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的贡献均已在论文中作了明确的说明。

作者签名：\_\_\_\_\_\_\_\_\_\_\_ 签字日期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

中国科学技术大学学位论文授权使用声明

作为申请学位的条件之一，学位论文著作权拥有者授权中国科学技术大学拥有学位论文的部分使用权，即：学校有权按有关规定向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅，可以将学位论文编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。本人提交的电子文档的内容和纸质论文的内容相一致。

保密的学位论文在解密后也遵守此规定。

□公开 □保密（\_\_\_\_年）

作者签名：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 签字日期：\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# 摘要

微软小娜是微软发布的全球第一款个人智能助理。在微软小娜应用程序中，日志数据记录了微软小娜自身的运行状态以及用户对微软小娜的使用情况。通过对日志数据的分析可以获取应用程序的运行状况、故障原因、用户需求和用户的访问特征等关键信息。

本文设计和实现了一个针对微软小娜产品的分布式日志收集、整理、计算和展示系统。微软小娜性能分析系统利用微软Aria平台进行日志收集和整理，并将日志数据保存在Cosmos分布式文件系统中；使用Scope脚本语言与C#语言相结合的开发方式进行产品性能计算；系统利用Windows服务器计划任务功能实现定时提交脚本和导入数据功能；最后系统使用ASP.NET框架开发网站，将微软小娜产品的性能数据以可视化的方式展示在网页上。本文按照软件工程的开发流程介绍系统的设计与实现过程，从系统的需求分析开始，经过总体设计、模块设计、系统实现和系统测试等一系列环节，最终完成微软小娜性能分析系统的设计与实现。

微软小娜性能分析系统为微软小娜的研发提供了数据上的支持，在微软小娜产品发布之前提供产品发布指标；在微软小娜产品发布之后提供各个功能模块的性能数据以及用户的使用情况，并根据用户习惯为微软小娜产品的下一步开发提供了建设性的意见。

**关键词**：性能数据，产品日志，微软小娜

# Abstract

Microsoft Cortana is the world's first intelligent personal assistant. The log files of Cortana record application runtime data and user operation data. Developer can get the application running status, the clue of bugs and user features by analyzing the raw log data.

The thesis designs and implements a system which could collect and calculate log data of Cortana on distributed platform and could display the performance metric on the website. The system uses Microsoft Aria Platform to collect logs, Cosmos File System to store logs, Scope scripting language and C# to compute performance metric, Windows server schedule task to submit scripts and import data, ASP.NET framework to build website which could display performance metric. This thesis describes the design and implementation of performance analysis system about Cortana by the process of software engineering. This thesis describes system requirements analysis, system design, module design, system implementation and system testing.

Performance analysis system provides performance metrics to support the development of Cortana. The system provides release metrics for developers before Cortana release. The system provides performance metrics and the usage report for each feature after Cortana release. According to user behavior the system could provide suggestions for the next generation development.

**Key Words**: Performance Metric, Product Log, Microsoft Cortana

# 目录

[摘要 I](#_Toc464396056)

[Abstract II](#_Toc464396057)

[第1章 绪论 1](#_Toc464396059)

[1.1 课题背景及选题意义 1](#_Toc464396060)

[1.2 国内外技术现状 2](#_Toc464396061)

[1.3 本文研究内容 4](#_Toc464396062)

[1.4 论文的组织安排 5](#_Toc464396063)

[第2章 关键技术介绍 6](#_Toc464396064)

[2.1 Cosmos平台和Scope脚本语言介绍 6](#_Toc464396065)

[2.2 ASP.NET MVC框架介绍 6](#_Toc464396066)

[2.3 通用Windows平台介绍 7](#_Toc464396067)

[2.4 数据库连接池介绍 9](#_Toc464396068)

[2.5 本章小结 9](#_Toc464396069)

[第3章 微软小娜性能分析系统需求分析 11](#_Toc464396070)

[3.1 日志收集需求分析 11](#_Toc464396071)

[3.2 日志计算需求分析 13](#_Toc464396072)

[3.3 自动化工具需求分析 15](#_Toc464396073)

[3.4 数据库和网站的需求分析 15](#_Toc464396074)

[3.5 非功能性需求分析 17](#_Toc464396075)

[3.6 本章小结 18](#_Toc464396076)

[第4章 微软小娜性能分析系统总体设计和模块设计 19](#_Toc464396077)

[4.1 总体设计 19](#_Toc464396078)

[4.1.1 设计目标和设计原则 19](#_Toc464396079)

[4.1.2 性能分析系统结构设计 20](#_Toc464396080)

[4.1.3 性能分析系统架构设计 23](#_Toc464396081)

[4.1.4 性能分析系统数据库设计 24](#_Toc464396082)

[4.2 模块设计 26](#_Toc464396083)

[4.2.1 日志收集模块设计 26](#_Toc464396084)

[4.2.2 性能数据计算模块设计 27](#_Toc464396085)

[4.2.3 自动化工具模块设计 30](#_Toc464396086)

[4.2.4 数据展示模块设计 33](#_Toc464396087)

[4.3 本章小结 35](#_Toc464396088)

[第5章 微软小娜性能分析系统实现 36](#_Toc464396089)

[5.1 系统平台的搭建 36](#_Toc464396090)

[5.2 日志收集模块实现 36](#_Toc464396091)

[5.3 性能数据计算模块实现 38](#_Toc464396092)

[5.4 自动化工具模块实现 40](#_Toc464396093)

[5.5 数据展示模块实现 46](#_Toc464396094)

[5.6 本章小结 54](#_Toc464396095)

[第6章 测试 55](#_Toc464396096)

[6.1 测试环境及方法 55](#_Toc464396097)

[6.2 功能测试 55](#_Toc464396098)

[6.3 性能测试 58](#_Toc464396099)

[6.4 性能数据分析 59](#_Toc464396100)

[6.5 本章小结 60](#_Toc464396101)

[第7章 总结与展望 61](#_Toc464396102)

[7.1 总结 61](#_Toc464396103)

[7.2 展望 61](#_Toc464396104)

[参考文献 63](#_Toc464396105)

[致谢 65](#_Toc464396106)

# 绪论

## 课题背景及选题意义

随着4G、Wi-Fi等移动通信技术的飞速发展，移动互联网行业的时代已经到来。作为用户进入互联网的重要入口之一，手机应用程序在移动互联网中已经占据了举足轻重的位置。根据iiMedia Research（艾媒咨询）2016年7月29日发布的《2016上半年中国手机桌面行业研究报告》，截止2016年第二季度中国智能手机用户已达6.308亿人[1]。庞大的用户基础推动了手机应用程序的快速发展。手机应用程序承载了各种便捷的移动服务，已经成为人们日常生活的一部分。手机应用程序的性能监控与应急处理已经受到移动互联网行业的重视。

为了更好的服务用户，了解用户的访问特征[2]及用户需求，了解手机应用程序的运行情况以及性能情况，手机应用厂商迫切需要对用户的使用情况进行分析研究。现在各大手机应用程序公司的普遍做法是记录用户使用情况，记录产品运行情况，然后根据产品运行数据和用户使用行为数据进行分析，进而改进产品给用户带来更好的体验。应用程序新版本在发布之前的性能测试也已经成为另一个非常重要的工作，通过产品的总体性能测试数据和各个模块分别的性能测试数据，开发人员可以快速发现并定位产品中存在的问题。另一方面，产品的性能数据也是衡量产品是否可以发布的重要指标。

产品日志是手机应用软件在运行过程中对自身运行状态进行实时记录的文件，通常这些文件还包括系统的操作、软件自身行为以及用户的行为[3]。这些数据通常用于软件故障排查和软件性能分析，在数据量不多时，日志的处理和分析由经验丰富的产品开发工程师人工完成，但是随着系统规模的扩大和用户数量的增加，日志的数量正以前所未有的速度增长，这种情况下继续采用人工的方式处理将是一件不可能完成的任务[4]。因此，为了高效的处理大规模日志，解决单机系统上的存储和处理瓶颈，提高处理效率，各大公司都在开发研究分布式日志管理系统和相应的软件性能分析系统[5]。

使用软件性能分析系统可以对大量的日志数据进行智能分析，自动计算产品性能，方便诊断和优化产品的缺陷[6]。软件性能分析系统可以量化的衡量整个产品的质量，可以通过分析产品中各个模块的数据，直观的得到每个模块的性能数据进而分析出产品的性能短板；可以通过分析产品不同流程不同模块的数据，快速发现和定位产品的问题，使开发工程师有的放矢；也可以通过分析用户数据，分析用户使用习惯，分析用户喜好，得出用户感兴趣的话题和功能，进而可以为产品下一步的改进提供建设性的意见。

微软小娜是微软发布的全球第一款个人智能助理，微软小娜可以帮助用户在电脑上查找资料、管理日历、跟踪程序包、查找文件、陪用户聊天，还可以讲笑话。使用微软小娜的次数越多，用户的体验将越来越个性化。微软小娜产品由语音识别、语义理解、语音唤醒和用户提醒等功能模块组成，其性能是指每个功能模块的成功率和延迟时间等数据。

因此，在微软小娜产品发布之初，应该分析产品性能，需要根据微软小娜特点开发性能分析系统，需要采用大数据处理的方法高效便捷的分析产品日志。性能分析系统主要目的是为微软小娜产品的发布和新功能的开发提供可以量化的性能指标，并且将这些指标实时的以可视化的方法展示给开发人员。

## 国内外技术现状

随着互联网的迅猛发展、系统规模的扩大和用户数量的增加，如今的互联网企业每天需要处理数以万计甚至百万千万的用户访问请求，用户的日志数量更是成几何倍数增加，这种高并发大数据的日志发送、接收和处理需要用到分布式软件系统（Distributed Software Systems，DFS）[7]。

分布式软件系统是基于通信网络互联的多处理机体系结构，并支持分布式处理任务的软件系统。分布式软件系统主要包括分布式操作系统、分布式程序设计语言及其编译系统、分布式文件系统和分布式数据库系统等。在日志收集中主要使用的是分布式文件系统[8]，分布式文件系统主要解决了数据的存储和管理问题：分布式文件系统可以隐藏底层实现，将多个地点或多个文件系统以一个文件系统的形态展示给用户，隐藏底层实现，组成一个统一管理的文件系统网络。

为了方便快速收集处理大规模并发数据，分布式日志收集系统已经得到了广泛应用。其中，Scribe是Facebook公司的开源日志收集处理系统，主要采用“分布式收集，统一处理”的可扩展高容错的方案，从各个日志源上收集日志并将日志存储到分布式文件系统上然后再统一处理[9]。

Chukwa是Apache公司的一个开源的大型分布式系统，主要用于日志的收集和分析[10]。Chukwa是构建在Hadoop的HDFS和MapReduce框架之上的数据采集分析框架，其继承了Hadoop的可伸缩性和鲁棒性[11]。该框架提供采集数据的客户端（Client），由客户端发送数据给服务器（Server），然后通过定期运行脚本分析数据。

Flume是由Cloudera公司提供的一个分布式可靠的海量日志采集、聚合和传输系统[12]。Flume支持在日志系统中定制各类数据发送方用于收集数据，同时对数据进行简单处理并且可以输出到各种数据接收方。

Aria是微软公司开发的Cosmos平台之上的日志收集系统，作为共享数据平台的一部分，Aria可以支持平均每秒5百万次日志收集。将日志导入Cosmos之后，采用Scope脚本更加详细的计算所需数据。Aria支持实时多维的大数据分析，用户可以创建展示页面，展示多维立体数据并且可以实时查看原始日志。

在大型软件系统中，由于系统环境设置和软件规模等因素，bug重现和调试都十分困难。百万用户级别的访问量每天会产生上百乃至上千兆字节大小的日志，为了能够高效快速的处理大量日志分布式大数据处理平台广泛应用于日志处理系统。微软主要采用Scope脚本语言对大规模用户日志进行处理。Scope语言是运行在Cosmos上的查询语言。Cosmos是微软研发的大数据处理平台。Cosmos主要有以下两大基本功能：

（1）使用分布式文件系统存储大量数据的功能；

（2）使用Scope语言处理大数据的功能。

Cosmos的主要特点如下：

（1）大规模数据处理能力，Cosmos支持PB级别的数据处理，拥有大量可信赖的计算机集群；

（2）Cosmos可以使处理大数据更加容易，Scope语言简单易学，允许我们在不懂得MapReduce细节[13]的情况下，使用简单SQL和C#[14]知识就可以进行大数据的运算处理；

（3）Cosmos也是一个共享数据系统（Shared Data Ecosystem），在工作中不同的组可以在系统提供的平台上相互共享数据。

SQL Server[15]是微软推出的关系型数据库，是一个可扩展的、高性能的、为分布式客户机和分布式服务器计算所设计的数据库管理系统，SQL Server可以提供了基于事务的企业级信息管理系统方案[16]。目前最新版本是2012年3月份推出的SQL SERVER 2012[17]。

互联网信息服务（Internet Information Services，IIS）是由微软公司提供的基于Microsoft Windows操作系统的互联网基本服务，用于文件传输、新闻服务、超文本传输服务、网页浏览和邮件发送服务。

## 本文研究内容

通过课题背景和国内外研究现状可以看到，研发一套性能分析系统是一项十分紧迫的任务，性能分析系统可以对产品所产生的日志数据进行智能分析、计算性能和诊断产品。

本文的目的就是开发一套针对微软小娜产品的日志性能分析系统，该系统使用微软公司的Aria分布式日志收集系统和Cosmos分布式文件系统来收集和存储日志，系统使用Scope脚本语言计算产品性能，系统使用SQL Server数据库存储产品性能数据，使用ASP.NET框架开发网站，使用IIS部署网站。小娜性能分析系统包括从用户使用产品产生日志开始，经过Aria收集日志并将日志导入到Cosmos分布式文件系统中，再使用Scope语言计算产品性能，最后将性能数据展示在网页上的整个日志采集及分析过程。本文的研究内容将从以下几个方面进行入手：

（1）Aria平台上日志的收集：日志收集主要通过Aria SDK实现，通过Tenant Token将日志数据与微软小娜关联起来，日志的内容除基本内容外扩展数据通过Key-Value的方式插入；在此方面还包含日志的设计，为了使日志计算更加合理使日志扩展性更强，设计了一系列措施保证数据的完整性、准确性和可扩展性。

（2）Cosmos平台上日志的计算：本系统采用Scope脚本语言进行性能的计算，主要计算各个模块的成功率和延迟时间。通过计算模块开始日志和结束日志之间的时间差计算模块延迟时间；计算模块结束日志数量占模块开始日志数量比例计算产品模块的成功率。其中，模块延迟时间不是简单的计算每个用户延迟时间的平均值，而是采用求分位数的方法将每个用户的延迟时间从小到大排序，取第75个、第90个、第95个百分位数作为模块的延迟时间[18]。

（3）自动化工具实现：为了能自动的提交脚本并将计算结果导入到数据库中，本系统实现了一个定时提交脚本并向数据库导入数据的工具，并且该工具可以对提交脚本和导入数据过程中可能出现的所有异常进行相应处理，保证工具的健壮性和灵活性。

（4）数据库和网站实现：微软小娜性能分析系统使用SQL Server数据库存储产品性能数据；采用ASP.NET MVC框架实现展示网站，将数据以可视化的图表和曲线图的方式展示出来，这样可以直观地看到产品各个功能模块的性能参数。

## 论文的组织安排

第1章，绪论部分，主要介绍了微软小娜性能分析系统的研究背景和项目介绍，介绍了目前日志收集系统和性能计算系统的国内外研究现状，以及本文研究的主要内容和论文组织架构。

第2章，关键技术介绍，主要介绍了微软小娜性能分析系统开发过程中需要采用的技术框架，涉及到的技术架构有ASP.NET MVC框架，涉及到的平台有Cosmos分布式文件系统平台和通用Windows平台，涉及到的技术有Scope脚本语言与C#结合的开发模式和数据库连接池技术。

第3章，微软小娜性能分析系统需求分析，主要介绍了性能分析系统的使用流程以及内部详细流程，根据性能分析系统的设计目标，给出了性能分析系统的具体需求，并且对各个模块进行了详细的需求分析。

第4章，微软小娜性能分析系统总体设计和模块设计部分，主要介绍了性能分析系统的业务逻辑，整理出系统的功能设计，对系统架构和数据库进行规划和设计；并且对系统的功能模块进行划分，针对每个功能模块进行设计，给出较为详细的时序图和描述，本章还对数据库表结构进行了详细的设计。

第5章，微软小娜性能分析系统实现部分，主要论述了系统的开发环境以及运行部署环境，对于系统每个子模块列出了详细的实现机制、具体的实现方法以及详细的实现类图。

第6章，测试部分，微软小娜性能分析系统主要是实现数据分析计算和展示功能。采用适当的测试方法，测试性能分析系统的运行情况以及各个子模块的运行情况。

第7章，总结与展望，对微软小娜性能分析系统的研究内容进行总结，给出了实现成果以及对未来的展望。

# 关键技术介绍

采用合理的软件开发框架和技术可以提高开发效率，增加微软小娜性能分析系统的鲁棒性和可扩展性。本章主要介绍开发微软小娜分析系统时使用的几个关键技术框架。

1. Cosmos平台和Scope脚本语言介绍

Cosmos平台是微软为存储和处理大规模数据而开发的，主要包括两大基本的功能：

（1）Cosmos文件系统（Cosmos File System，CFS）可以支持PB级别数据计算，是一个具有存储大规模数据能力的分布式文件系统；

（2）使用Cosmos平台提供的Scope脚本语言可以处理存储在Cosmos文件系统中的数据，Scope脚本语言可以使我们非常容易的处理大规模数据，只需要基本的SQL和C#语法就可以使用Scope语言，而且现在可以使用Visual Studio方便的开发Scope程序。

Cosmos平台的大规模数据存储和处理能力主要体现在以下几个方面：

（1）Cosmos平台大约有11万台处理器分布在4个不同的物理集群上；

（2）Cosmos文件系统存储了1.2EB的数据（）；

（3）Cosmos平台每天可以运行10万多个Scope程序。

Cosmos也是一个数据共享系统，不同的团队可以通过Cosmos平台共享数据集。Scope是Cosmos平台的查询脚本语言，它的许多基本概念和句法表达式都和SQL类似，这样使得程序简单易写，而且Scope还可以调用C#函数，这使得Scope更加的灵活和强大。在Visual Studio中，用户可以使用Scope的开发插件方便快捷的编写程序和提交到Cosmos平台上运行。

1. ASP.NET MVC框架介绍

ASP.NET[19]是.NET Framework的一部分，它可以使嵌入网页中的脚本在因特网服务器短执行，它也可以动态的在Web服务器上被创建[20]。ASP .NET的前身是动态服务器网页（Active Server Pages，ASP）技术，与ActiveX数据对象 （ActiveX Data Objects，ADO）1.0在IIS 2.0上首次推出，在IIS 3.0时为大家广泛熟知，逐渐成为网页应用程序的几个主要开发工具之一[21]。ASP.NET具有很高的灵活性，支持三种开发框架：Web 页面（Web Page）、模型-视图-控制器（MVC）[22]和Web 窗体（Web Forms）。

MVC框架将应用程序分成3个组件：模型（Model）、视图（View）和控制器（Controller）。模型包含应用程序核心功能和数据，在应用程序中负责处理数据逻辑；视图在应用程序中负责数据显示；控制器处理输入，在应用程序中负责处理用户交互[23]。

模型封装了应用程序的功能和数据[24]，不对用户可见，模型和视图相互独立；输出表示或者输入方式与模型无关，模型与控制器独立；用户不能直接操作应用程序的功能和数据，只能通过控制器进行，控制器是模型和视图的桥梁。

视图展示应用程序界面信息，每个视图都会对应一个方法，通过这个方法展示数据，每个视图也会对应一个状态查询函数，目的是使得视图和模型一致，视图通过状态查询函数从模型获取数据，如定时刷新。

每个视图都与一个控制器相互关联，视图与控制器相互对应。当用户请求到达时，控制器负责接受事件，并翻译成对模型或者视图的请求。如果控制器处理请求的方式与模型相关，则控制器需要在模型中注册一个变更函数，用户仅仅通过控制器与系统交互。

1. 通用Windows平台介绍

通用Windows平台（Universal Windows Platform，UWP）应用是一种基于通用Windows平台生成的可在所有Windows上运行的应用，它在Windows 8操作系统中被首次引入[25]。Windows 8引入了Windows运行时，它的设计初衷是成为通用应用程序体系结构。在Windows Phone 8.1后，Windows运行时在Windows Phone 8.1与Windows之间仍保持一致。Windows 10引入了通用Windows平台推动了Windows运行时模型的发展。

通用Windows平台是一个可在所有运行Windows 10的设备上使用的平台。面向通用Windows平台的应用程序不仅可以调用对所有设备均通用的Windows 运行时接口，还可以调用某些特定设备的接口。UWP对于不同Windows设备可以提供相同的API图层。

UWP的核心是使用户在所有设备上的体验相同，并且希望使用现有的最方便最高效的设备完成任务。

因为通用Windows平台应用可在各类具有不同外形规格和输入方式的设备上运行，所以用户希望针对每个设备进行定制特有的输入方式，而且希望每个设备都能具有独特功能。

开发人员将设备中特殊的API添加到统一的API图层，这样开发人员可以编写代码以特殊条件访问这些特殊的API，这样应用程序就可以在其他设备上运行并提供不同的体验。

通过Windows 10操作系统，用户可以更加轻松地开发UWP应用，只需一个API集合、一个应用和一个应用商店，即可访问所有Windows 10设备包括电脑、平板、手机、Xbox、HoloLens等。对不同大小屏幕以及各种交互模型（无论是触控、鼠标、键盘还是游戏控制器）的支持也越发方便[26]。UWP应用的主要特性如下：

（1）开发人员的目标设备是整个Windows系列，而非某个操作系统。

（2）UWP应用使用AppX格式进行打包和部署，这提供了值得信赖的安装机制，确保应用可以无缝进行部署和更新。

（3）应用商店中的应用程序适用于所有Windows设备。应用开发人员可以向Windows应用商店提交应用，可以选择应用使用的设备，可以在所有设备系列上运行也可以在所选设备上运行。

（4）在整个Windows系列上有一套相同的应用程序编程接口。所有Windows设备系列均使用相同的接口，这样开发出来的应用将可以在任何Windows 10设备上运行。

（5）在特殊的设备上使用扩展SDK可以为应用增添色彩。扩展SDK可以为每个设备系列添加专用API，如果应用面向特定的设备系列，开发人员就可以使用这些API。

（6）UWP应用具有自适应输入控件，它们与多种输入类型（如键盘、鼠标、触摸、笔等）配合良好。UWP应用的UI元素使用有效像素，它们会基于设备的屏幕像素数自动自行调整。

1. 数据库连接池介绍

微软小娜性能分析系统中网站使用数据库连接池来管理维护数据库的连接。建立一个数据库连接是一件非常耗时耗力（消耗时间消耗资源）的事情，连接到数据库服务器通常由几个步骤组成:建立物理通道；与服务器进行首次连接；分析数据库连接串信息；进行用户身份验证等，这几个步骤都需要较长的时间才能完成。

当应用程序访问次数数量很大的时候，连接相同数据库的连接可能会被反复地建立和回收，这样可能引起数据库效率下降，又因为大部份的应用程序都是使用一个或几个固定的连接配置，因此本文采用ADO.NET[27]框架，并使用该框架提供的数据库连接池来优化和管理数据库。

数据库连接池存放了一定数量的与数据库服务器相连的物理连接。当程序需要连接数据库服务器的时候，应用程序就会检查连接池中是否有空闲连接，如果发现数据库连接池中有空闲连接，就会直接使用这个连接；当应用程序需要关闭数据库连接时，数据库连接池会判断当前连接数是否小于最小连接数，如果小于数据库将连接回收到数据库连接池中，如果大于则释放相关资源并回收连接。这样就可以大大减少连接数据库的开销，从而提高网站的性能。高效的使用数据库连接池一般需要遵循以下几个原则：

（1）在最晚的时刻申请数据库连接，在最早的时候释放数据库连接；

（2）关闭连接时先关闭相关用应用程序定义的事务；

（3）确保并维持数据库连接池中至少有一个打开的连接；

（4）尽力避免池碎片的产生，包括集成安全性产生的池碎片以及使用许多数据库产生的池碎片。

1. 本章小结

本章主要介绍微软小娜性能分析系统的设计开发实现过程中，所使用的关键技术框架及其特点，包括Cosmos平台和Scope脚本语言、ASP.NET MVC框架、通用Windows平台应用、数据库连接池等。采用Cosmos平台和Scope脚本语言能够方便高效的计算大规模日志数据；采用ASP.NET MVC框架搭建网站可以使程序容易维护，增强程序可扩展性，能够进行单元测试，保证功能的实现；采用通用Windows平台应用可以开发一套适用所有Windows平台的应用程序；采用数据库连接池可以大大减少网站连接数据库时引起的开销，从而提高网站的性能。接下来对微软小娜性能分析系统的需求进行分析。

# 微软小娜性能分析系统需求分析

详细梳理微软小娜性能分析系统的内部流程，能够全面地指出微软小娜性能分析系统的具体需求；有针对性地梳理系统的业务处理模块，能够归纳总结出系统的功能性需求；基于对系统性能提升的考虑，可以对非功能性需求提出要求。根据微软小娜性能分析系统的主要功能将该系统分为4个部分包括Aria上日志的收集、Cosmos上日志的计算、自动化工具实现、数据库和网站实现，下面将详细针对每一部分实现其需求分析。

1. 日志收集需求分析

微软小娜性能分析系统的第一步就是获取用户操作数据和产品运行数据，获取产品客户端的日志，即日志的收集。当用户操作产品时，开发人员希望记录产品的版本号，产品的应用平台，用户操作的具体功能，用户操作的准确时间，用户所在地区，用户使用的语言以及区分日志的编码字段等关键信息。根据日志收集的需求设计日志结构如表3-1所示：

表3-1 微软小娜日志结构

|  |  |
| --- | --- |
| 属性 | 值 |
| Event Name | CoA App Load |
| Impression ID | 17b84a2c-e4e7-4925-8ab3-9c24d5ebd368 |
| App Version | 1.0.0.662-release |
| OS Version | Android 4.4.4 |
| Time Stamp | 188469142 |
| Language | En-US |
| Key | Value |
| … | … |
| Key | Value |

其中每个字段的含义如下：

Event Name：该字段主要为区分产品中不同模块不同功能的日志，目的是使日志的计算更加方便。产品初始加载日志的Event Name可以设为CoA App Load，闹钟模块日志该字段可为CoA Alarm。

Impression ID：该字段为日志序列号，主要是为区分不同用户不同操作的日志数据，便于性能数据的计算，如果两条日志的Impression ID相同那么就表示这两条日志发生在同一客户端。例如要计算产品加载的时间延迟就需要在数据集中获得Impression ID相同的产品加载开始日志与加载结束日志，通过这两条日志的时间差就可以计算出该用户使用产品时产品加载的时间。Impression ID的编码可以采用通用唯一识别码（Universally Unique Identifier，UUID）标准，其目的是使所有日志都有唯一辨识的资讯，UUID标准可以保证编码的唯一性[28]。

App Version：该字段记录当前操作发生时产品的版本号，主要目的是区分不同版本产品性能，进行纵向对比。版本号的具体规定为正式版本以-release结尾，测试版以-dev结尾，微软小娜产品的版本号码由4个十进制数组成，数之间以点号分隔，如1.0.0.662-release。

OS Version：该字段记录当前操作发生时手机系统的版本号，如Android 4.4.4，主要目的是对比不同手机系统之间的性能差异，使开发人员可以针对某一操作系统进行测试优化。

Time Stamp：该字段记录当前日志的生成时间，用来计算产品模块或功能的延迟，精确到毫秒。在此Time Stamp是指格林尼治时间1970年01月01日00时00分00秒起至现在的总毫秒数。

Language：该字段存放用户使用的语言和所在地区，产品主要的字段值为：En-US（英语-美国）、En-IN（英语-印度）、Zh-CN（汉语-中国）、Ja-JP（日语-日本）。

Key-Value：该字段存放开发人员为各个模块定制的参数，这些数据以Key-Value的方式存放，可以细分产品的子功能，更加精确的计算各个子功能的性能参数。

整理好日志的基本格式之后，需要对日志的发送和收集储存进行分析。用户的日志收集采用Aria分布式日志收集平台实现，通过在客户端加入Aria SDK，客户端就可以主动发送用户日志到Aria服务器，然后Aria服务器将各个产品的日志数据进行整理，并把微软小娜的日志数据储存在Cosmos分布式文件系统中供下一步计算使用。

1. 日志计算需求分析

在整个性能分析系统中，最关键最重要的模块就是日志计算模块，因为日志计算的准确度直接关系到性能数据的准确度，是整个项目的核心。



图3-1 微软小娜性能分析系统的性能指标

微软小娜性能分析系统需要分析的性能指标结构如图3-1所示，分为服务器性能和客户端性能两部分，服务器端性能主要包括HTTP连接建立的成功率和连接延迟时间、Notification成功率；客户端性能主要包括语音识别功能、语义理解功能、APP加载过程、语音唤醒功能、数据上传功能、MSA账户登录过程等功能的成功率和延迟时间。

微软小娜客户端的性能数据需要计算不同版本号、不同地区、不同网络环境、不同操作系统、不同功能模块的性能数据，并对其进行纵向比较，因此，在日志计算的过程中需要对以上需求进行分别计算。其中各个需求的划分方法分别为：版本号为每次产品发布时的版本号，具体规定为正式发布版本以-release结尾，开发测试版以-dev结尾，各个功能模块的测试版以-rd结尾，版本号码由4个十进制数限定，数之间以点号分隔，如1.0.0.662，随着发布版本增加版本号会逐渐增大；不同的网络主要是指网络运营商和网络制式的区分，在中国大陆地区网络运行商主要有中国移动、中国联通和中国电信，在美国和欧洲网络运营商主要有AT&T、Sprint、T-Mobile、Verizon Wireless、MetroPCS和Cricket Communications，网络制式主要有HSPA+、LTE、HSDPA、HSPA、UMTS、CDMA-eHRPD、TD-SCDMA、EDGE、GPRS，计算性能数据时，系统会根据网络制式将其划分为4G、3G、2G以及Wi-Fi网络，以便于计算；因为各个地区的语言和习惯不同，微软小娜产品会根据地区不同对产品做出一些功能上的调整，因此需要对不同的地区分别计算产品性能，现在需要计算地区有美国、印度、日本和中国4个地方；在操作系统上系统主要区分iOS和Android，后面可能会有WP等其他操作系统的需求。

微软小娜性能分析系统有两大主要性能指标：成功率和延迟时间。成功率是指每个功能模块在没异常情况下正常运行到结束的概率；延迟时间是指产品的每个功能模块从开始加载到结束的时间，需要精确到毫秒，最关键的是功能模块的延迟时间不应该简单是每个用户延迟时间的平均值，而应该采用求分位数的方法将每个用户的延迟时间从小到大排序，取第75个、第90个和第95个百分位数作为模块的延迟时间，因为求平均值的方法可能会引入较大误差，使用这种计算方法可以降低延迟时间的误差（假设第75个半分位数的延迟时间为2000毫秒，表示有75%的用户的延迟时间小于2000毫秒）。根据产品的特点、产品的功能分析，需要计算的场景举例如下：

语音识别（Speech Recognition，SR）：指从用户点击麦克风录音开始到语音识别模块识别完全部文字的过程，需要计算语音识别模块的成功率及其延迟时间，如果用户主动取消了语音识别操作，也认为语音识别模块失败。

语义理解（Language Understanding，LU）：指从语音识别完全部文字到客户端收到HTTP回复语义理解结果的过程，需要计算语义理解模块的成功率以及延迟时间，如果用户主动取消了语义理解操作，也认为语义理解模块失败。

Proactive Page Load：客户端主页面加载场景，指从发出加载页面请求开始到主页面被全部加载完成的过程，需要计算其成功率以及延迟时间，如果用户主动取消了页面加载操作，也认为页面加载模块失败。

App Load：产品启动，指从用户在桌面点击小娜图标开始到用户可以使用麦克风或搜索框搜索的过程，需要计算其成功率以及延迟时间，如果用户主动取消了产品启动操作，也认为产品启动模块失败。

日活跃用户量（Daily Active User，DAU）：指微软小娜每天的活跃用户量，需要计算每天微软小娜产品总体及各个功能、各个模块的分别活跃用户量，也包括每日新用户数量已经新用户次日存留用户量。

1. 自动化工具需求分析

实现自动化工具的目的是自动化工具可以每天定时提交脚本并自动将已经计算的性能数据导入到数据库中，因此该工具主要有定时提交脚本和将性能数据导入到数据库这两大需求。Cosmos分布式文件系统中的日志数据会有延迟时间，一般的延迟时间为2天，但是也可能会有更多天数的延迟，因此自动化工具需要记录两个日期，分别是已经将脚本提交的日期，和已经将性能数据导入到数据库的日期。这样做的目的是防止重复计算降低系统性能，防止重复数据导入数据库。系统每次运行之前，需要检查之前脚本的提交日志和导入数据的日志，以便重新提交延迟的产品日志。此外还可能会有某一天或一段时间的性能数据因为日志问题需要重新计算，自动化工具需要有对指定日期或指定时间段的数据进行重新计算，并把已经导入数据库的数据删除。

自动化工具应该可以简单的被应用于其他程序，因此需要有动态配置脚本优先级、脚本路径、数据下载路径以及数据库名称以及表名称的功能。除此之外，自动化工具还应该每天定时自动运行，这样可以最大程度上节约开发人员的时间。最重要的是自动化工具需要具有异常处理、上下文恢复的功能，因为在提交脚本时可能会由于超时，脚本编译错误，源文件源目录不存在等问题出现异常，这时应该把所有情况恢复到工具未运行的情况，并需要记录异常日志，特别的，当数据导入数据库发生异常时，自动化工具需要将已经导入的部分数据删除，并记录异常日志。当有异常发生时，自动化工具应该有自动提醒开发人员的功能，可以使开发人员快速收到异常日志或错误记录，以便快速发现问题解决问题。

1. 数据库和网站的需求分析

计算出的产品性能数据都会被保存在数据库中以供网站读取，因此在数据库的需求分析阶段需要确定基本的表结构，数据库主要存储的数据可以分为三类：功能的成功率、功能的延迟时间和功能的使用人数。

在数据库保存产品功能的成功率时，需要存储日志的产生时间（精确到天），功能的名称，功能的子名称，产品的版本号，手机的网络状况，手机的系统版本号，用户的语言和地区，以及功能的使用次数，功能被成功使用的次数和功能的成功率。在讨论延迟时间的需求时需要定义几个指标PLT75、PLT90、PLT95，PLT75、PLT90、PLT95的含义是假设PLT75等于2000毫秒，表示有75%的用户的延迟时间小于2000毫秒，其计算方法是将每个用户的延迟时间从小到大排序，取第75个、第90个和第95个百分位数即为PLT75、PLT90和PLT95的值。在数据库保存产品功能的延迟时间时，需要存储日志的产生时间（精确到天），功能的名称，功能的子名称，产品的版本号，手机的网络状况，手机的系统版本号，用户的语言和地区，功能的使用次数以及PLT75、PLT90和PLT95的值。在存储功能的使用人数时，需要存储日志的产生时间（精确到天），产品的使用人数，产品当天的新用户，产品昨天的新用户在今天的使用人数，除此之外还要存储各个功能模块的使用人数，此功能当天的新用户，此功能昨天的新用户在今天的使用人数。

网站功能性需求为：用户可以查询每天的产品性能数据，用户可以选择不同的网络数据（Wi-Fi/4G/3G/2G）、不同的产品发布市场（Zh-CN/En-US/En-IN/Ja-JP）、不同的产品系统平台（Android/iOS）查看相关功能模块的性能数据，用户可以指定功能模块查看其几个月之内的性能变化趋势图。根据数据库存储的三类数据：功能的成功率、功能的延迟时间和功能的活跃用户量。网页需要划分为三个区域，分别展示功能模块的成功率、延迟时间和活跃用户量。网站需要根据成功率和延迟的数据大小对数据标注不同的颜色，以求更直观的看出产品各个功能模块的性能，具体标准为当成功率高于99%时标为绿色，当成功率高于90%时标为黄色，当成功率低于90%时标为红色；当延迟时间小于1500毫秒时标为绿色，当延迟时间小于2500毫秒时标为黄色，当延迟时间大于2500毫秒时标为红色。绿色数据表示该功能模块运行良好，黄色数据表示该功能模块性能需要注意，红色表示该模块性能出现较大问题，当某个模块的性能参数出现较大问题时，网站应该又给开发该功能的开发人员邮件提醒的功能。为了纵向比较产品的性能，当用户点击任何一个数据时，网站需要展示相同功能，相同的语言和地区，相同的手机系统，相同的网络状况下此数据两个月以来变化的趋势图。对于每个大功能模块的成功率和延迟，需要画出饼状图展示每个子功能模块的成功率和延迟时间；对于每个功能模块的失败原因进行分析，统计不同失败原因导致功能失败的次数，并画出饼状图。网站的用例图如图3-2所示。



图3-2网站用例图

1. 非功能性需求分析

微软小娜性能分析系统的非功能性需求主要包括两个方面：性能需求和扩展性需求。

因为开发人员要关注当前版本产品的性能情况，因此需要整个分析系统应支持小时不间断工作；如果发现产品性能下降，开发人员需要查看用户的原始日志发现问题，因此用户的原始日志应该永久保存；分布式文件系统中产品的性能数据应保存3个月；数据库中产品的性能数据应永久保存；可扩展性指性能分析系统应支持业务功能的扩展，当增加模块或减少模块时，性能分析系统应该自动计算新增加的模块，并在网页上自动增加相关表格支持新模块的性能数据展示，这样能有效的减少后续系统开发和投入成本。

1. 本章小结

本章从微软小娜性能分析系统的业务流程入手，介绍了性能分析系统的详细流程，引出了微软小娜性能分析系统在计算和展示产品的性能数据时所要解决的问题，并给出了具体解决方案。在功能性需求中，从Aria上日志的收集、Cosmos上日志的计算、自动化工具实现、数据库和网站实现四个功能模块入手，对业务的需求进行详细剖析；在非功能性需求中，对微软小娜性能分析系统的性能需求和扩展性需求进行详细的说明。详细的需求分析是微软小娜性能分析系统的基础环节，保证设计有序展开及最终的实现结果能够满足要求。

# 微软小娜性能分析系统总体设计和模块设计

本章主要介绍微软小娜性能分析系统的总体设计和模块设计，将需求分析得到的系统功能性和非功能性需求换为软件结构和数据结构，对每个功能模块功能进行说明，并描述了任务调用流程，为性能分析系统相关模块功能的开发实现打下坚实的基础。

1. 总体设计

总体设计将微软小娜性能分析系统按功能进行模块划分、建立模块的层次结构及调用关系、确定模块间的接口及网站界面，除此之外还会对数据结构进行设计，包括性能数据特征的描述、确定数据的组成结构、以及数据库的设计。

1. 设计目标和设计原则

微软小娜性能分析系统是为了更有效的实现快速化、规范化、流程化的分析微软小娜产品而建立的分析系统。通过微软小娜性能分析系统，可以有效的收集存储大规模日志数据，可以快速有效的计算产品性能以及更加直观的看出产品性能数据。这样可以极大的提高性能分析工作的效率，提升微软小娜产品的研发能力和综合实力。

微软小娜性能分析系统各个子模块间应该能够相互独立的工作也可以动态的进行组合，根据不同的使用场景系统可以提供不同的操作项。这样既可以增强系统的可维护性，又增加了系统的可扩展性。系统要达到的目标就是提供一个可靠、稳定、快速获得产品性能数据的性能分析系统，为了达到这个目标，制定如下设计目标：

（1）使用模块化、标准化和参数化的设计标准，保证系统的灵活性、实用性、稳定性、开放性、先进性和可扩展性，使任何新的业务需求和产品需求都可以快速实现；

（2）微软小娜性能分析系统需提供数据统计和决策支持的功能，帮助开发人员快速获得产品性能参数和市场投放效果，并根据用户行为用户习惯完善产品的功能；

（3）微软小娜性能分析系统应该具有一定的容错能力，在日志数据出现异常的情况下也不会崩溃，而是可以出现合理的解决提示。

为了实现一个可靠、稳定、可以快速获得产品性能数据的性能分析系统，特制定如下设计原则：

（1）可靠性原则：对于性能分析系统来说，数据源的准确性和完整性是整个系统的基础，对于整个性能分析系统至关重要，这涉及到后面的计算环节，如果用户日志数据不准确、不完整将直接影响到产品性能数据的准确性，影响到整个系统的可信度，这就要求日志的收集和存储需要有高度的可靠性；

（2）健壮性原则：系统需保证无论在何种优化和调试情况下，无论发生什么异常情况，都要保证系统的稳定运行，获得的数据必须稳定、可靠、可信赖，要保证数据库数据的准确性，所有存储在数据库中的数据必须是准确的，可以作为评估产品指标的；

（3）先进性原则：先进性需要系统能够最大程度适应业务需求变化、产品需求变化以及技术发展变化，因此，在设计系统时需要采用先进的设计模式和开放的技术框架，使用先进的管理方式，保证系统的先进性；

（4）可扩展性原则：在设计系统时需要采用参数化、规范化的配置方式，模块化的设计理念使系统的各项指标都可以灵活设置，使系统具有良好的可移植性，另外数据存储结构的设计也需要满足先进、可扩展、易维护的要求。可扩展性是评价一个系统性能优劣的重要指标之一，良好的可扩展性必然可以提高系统的可维护性，这是软件项目开发过程中很重要的部分。

1. 性能分析系统结构设计

微软小娜性能分析系统涉及到众多子模块的交互，Aria收集日志平台与Cosmos分布式文件系统存储模块的交互，Cosmos分布式文件系统存储模块与自动化运行脚本工具的交互，自动化运行脚本工具与数据库模块的交互，网站展示模块与数据库模块的交互。根据微软小娜性能分析系统模块的功能得到系统结构设计图如图4-1所示。

（1）日志模块：负责收集用户日志、整理用户日志并将用户日志数据保存在Cosmos分布式文件系统中；



图4-1 微软小娜性能分析系统结构设计图

（2）性能数据计算：根据业务需求和产品需求计算日志数据获得产品的性能参数，主要包括功能模块的成功率，功能模块的延迟时间和功能模块的活跃用户量；

（3）自动化工具：自动化工具需要每天定时提交脚本并计算日志数据，然后将计算的产品性能数据保存在数据库中，在数据和工具出现异常时，自动化工具需要对异常进行自动处理；

（4）网站：网站将保存在数据库中的性能数据展示在网页上，根据不同的业务需求以不同的方式展示不同的数据。

为了提高微软小娜性能分析系统功能的扩展性，在系统以功能划分的基础上，微软小娜性能分析系统总体设计采用平台化设计思想。微软小娜性能分析系统的活动图如图4-2所示，客户端先产生日志，然后系统收集日志并用Cosmos文件系统存储日志，接下来使用Scope脚本计算功能模块的成功率、延迟时间和活跃用户量，然后将计算的性能数据保存在数据库中，最后网站从数据库中读取性能数据并展示在网页上。



图4-2 微软小娜性能分析系统活动图

网站需要包含以下几个页面：产品发布指标页面、网络分类后的性能指标页面、通信顺序进程指标页面。产品发布指标页面：此页面是在产品发布之间建立，目的是为产品的发布提供性能参数。页面主要展示产品三个部分的性能参数语音触发部分，主动触发部分和其他部分，每一部分都包含成功率和延迟两块内容。语音触发部分是指用户主动语音输入所引发的相关功能，如语音识别、语义理解、数据请求等，具体的产品功能有设定闹钟、询问天气、打电话、发短信、查看日历等功能，针对每种功能分别分析他们的成功率和延迟；主动触发部分是指用户直接点击小娜按键和小娜笔记本中的相关功能的性能参数，主要的性能参数也是成功率和延迟；其他部分主要考察的部分有：MSA账户注册及登录、用户坐标定位功能、用户历史数据上传功能、用户设定提醒上传功能、小娜语音唤醒功能等一系列功能的成功率和延迟。而且再次页面中如果用户想查看某个数据的趋势图，只用点击相关数据就可以看到2个月以来相关数据的折线图。

网络分类后的性能指标页面：此页面是在产品发布之后为了查找产品性能与网络制式以及网络运营商之间的关系建立的页面。在此页面中我们将每个性能的数据以表格和折线图的方式给出，表格展示每个功能在不同网络运营商和网络制式下的性能数据，折线图展示每个功能两个月以来在不同网络运营商和网络制式下的性能数据，分析趋势，而且可以选择展示为多条曲线还是每个网络制式分别展示，也可以选择不同网络制式进行比较。

通信顺序进程页面：主要指从用户点击语音按钮到语音说话开始，然后语音流传输，再然后语音识别模块返回第一个字符，再到用户讲完语音输入结束的一系列过程，其中从用户开始讲话到讲话结束的过程也是HTTP连接的过程。在此过程中，系统主要针对以下几个方面进行性能展示：从用户语音输入到语音识别模块检测到语音整个过程的成功率和延迟，从语音识别模块检测到语音到返回第一个字节的识别结果整个过程的成功率和延迟，从语音识别模块返回第一个字节的识别结果到语音识别模块识别完所有的语音整个过程的成功率和延迟，还有整个HTTP连接过程的成功率和延迟。在通信顺序进程相关的日志中，系统会记录会话失败的原因，在展示数据的时候我们把每天失败的原因以饼状图的形式展示出来，可以直观的看到通信顺序进程连接中最多的失败原因，并加以改进。

1. 性能分析系统架构设计

在对微软小娜性能分析系统进行架构设计时，本文采用模块化的架构设计方法，将性能分析系统主要分为以下几个模块：日志收集模块、性能数据计算模块、自动化工具模块、数据展示模块。最主要的几个子模块架构设计如下：

（1）性能数据计算模块：本模块采用Scope + .NET框架实现，采用面向过程的实现方法，分析出解决问题所需要的步骤，然后用函数把这些步骤一步一步实现，使用的时候一个一个依次调用。

（2）自动化工具模块：本模块采用.NET + XML+ [Multithread](https://www.baidu.com/link?url=8DKEiKfGoKkVZmhQXoe9WycO1t475c9EPfUKesaQgSFJfFb7-GmVEb6CgD95dzc837gBZRpAAAj7BrrGpnzHg4YqY7wvrkGvX77GhrrZ3-C7scBAXnrmVKLEwnFR5FCj&wd=&eqid=d9884b5c00009d950000000357a485ad)框架实现，自动化工具对每一个单独的任务都起一个线程单独执行，因为每次提交脚本执行都会耗费较长时间，采用多线程框架可以有效的节约时间，可以使程序的响应速度更快；除此之外还将脚本参数写在XML文件中，当添加脚本删除或脚本更改路径时，只需要更改XML文件而不需要重新部署，可以显著提升模块的可扩展性；

（3）数据展示模块：本模块采用ASP .NET + MVC框架实现，模型包含核心功能和数据，是数据展示模块中用于处理应用程序数据逻辑的部分；视图部分即网站页面负责向用户显示微软小娜功能模块的性能数据；控制器处理处理用户交互，根据用户输入展示出不同的性能。

结合微软小娜性能分析系统采用的技术架构，要完成一个优秀合理的性能分析系统本系统设计上分三层，即数据计算层、任务处理层和数据访问层，每层都能完善自身的功能，最终将每一层整合起来，就可以完成一个优秀的性能分析系统。数据计算层是整个性能分析系统的核心，根据业务需求的不同计算不同的数据；任务处理层是整个系统的中枢，起到承上启下的作用，用户触发任务，处理层进行具体业务处理，读取并展示相应的数据；数据访问层，采用数据库连接池技术，减少数据库连接时间，而且该层实现了系统中设计的数据访问，提高了代码重用性，消除了上层对数据源的直接依赖，隐藏了实现细节，降低了模块间的耦合性。

1. 性能分析系统数据库设计

微软小娜性能分析系统采用SQL Server数据库。SQL Server是微软推出的一个关系型数据库，是一个可扩展的、高性能的、为分布式服务器计算所设计的数据库管理系统，提供优秀的并发控制方案，具有良好的运行效率，有利于提高分析系统的运行效率，同时Visual Studio完美支持SQL Server的开发工作，开发人员可以方便的在Visual Studio上查看、调试和管理数据库。微软小娜性能分析系统的表结构说明，如表4-1所示：

表4-1 微软小娜性能分析系统的表结构说明

|  |  |  |
| --- | --- | --- |
| 表名 | 描述 | 功能 |
| AvailCoA | Android版微软小娜成功率 | 保存Android版微软小娜每个功能模块的使用次数和成功率 |
| AvailCoAByNetwork | Android版微软小娜按照网络分类后的成功率 | 保存Android版微软小娜按照网络分类后每个功能模块的使用次数和成功率 |
| LatencyCoA | Android版微软小娜延迟时间 | 保存Android版微软小娜每个功能模块的使用次数和延迟时间 |
| LatencyCoAByNetwork | Android版微软小娜按照网络分类后延迟时间 | 保存Android版微软小娜按照网络分类后每个功能模块的使用次数和延迟时间 |
| UsageCoA | Android版微软小娜用户量 | 保存Android版微软小娜每天的活跃用户量和每天的新用户量 |
| AvailCoI | iOS版微软小娜成功率 | 保存iOS版微软小娜每个功能模块的使用次数和成功率 |
| AvailCoIByNetwork | iOS版微软小娜按照网络分类后的成功率 | 保存iOS版微软小娜按照网络分类后每个功能模块的使用次数和成功率 |
| LatencyCoI | iOS版微软小娜延迟时间 | 保存iOS版微软小娜每个功能模块的使用次数和延迟时间 |
| LatencyCoIByNetwork | iOS版微软小娜按照网络分类后延迟时间 | 保存iOS版微软小娜按照网络分类后每个功能模块的使用次数和延迟时间 |
| UsageCoI | iOS版微软小娜用户量 | 保存iOS版微软小娜每天的活跃用户量和每天的新用户量 |

完善优秀的系统表设计是微软小娜性能分析系统的基础，优秀的系统表能够向系统提供可靠的存储、维护、检索数据功能。微软小娜性能分析系统数据库主要涉及到的信息包括，各个功能模块的用户量，各个功能模块的使用次数，各个功能模块的用户次日存留率，各个功能模块的成功率和各个功能模块的延迟时间。在实现微软小娜性能分析系统数据库的过程中要遵循以下原则：保持规范化，满足范式，减少重复数据；设计合理，结构优化；满足性能分析系统的业务扩展性。

1. 模块设计
2. 日志收集模块设计

日志收集模块的主要功能为当用户操作微软小娜客户端或客户端对其运行状态进行记录后，客户端将记录的日志信息发送到Aria日志收集平台，然后Aria日志收集平台将分类整理后的日志保存在Cosmos分布式文件系统中。

在日志收集模块中主要包含三个主体：微软小娜客户端、Aria日志收集平台和Cosmos文件系统，微软小娜客户端将日志发送到Aria日志收集平台，Aria日志收集平台将日志是否发送成功的消息返回给微软小娜客户端，最后Aria日志收集平台将分类后的日志保存在Cosmos文件系统中。日志收集模块时序图，如图4-3所示。



图4-3 日志收集模块时序图

当用户对微软小娜客户端进行操作或客户端对其运行状态进行记录后，客户端会将日志数据发送给Aria日志收集平台，然后日志收集平台会初步分类、整理日志并将日志数据保存到Cosmos分布式文件系统中。因为每次发送日志都需要客户端和Aria日志收集平台建立连接，如果客户端每次产生日志，就马上将日志发送到日志收集平台就会造成资源浪费的情况，而且会使手机的电量急剧下降，因此客户端只会在固定的几个时间发送日志，最关键的是当日志发送失败时，回调函数会通知微软小娜客户端日志发送失败需要重新发送日志数据。因为Aria日志收集平台不仅仅收集微软小娜的日支数据，所以Aria还需要将日志进行分类整理才能将日志保存在Cosmos文件系统中。

当微软小娜产生日志数据后，产品后台不会立即将日志数据发送给Aria日志收集平台，而是当日志累计一定得规模触发上传日志功能，或者以其他方式触发产品后台上传日志。这样可以减少产品与服务器的交互次数，节约客户端电量。在用户离线操作时也可以记录用户数据，在用户上线时上传日志。由此产生的日志延迟时间并不会对系统的性能数据计算产生影响。

1. 性能数据计算模块设计

性能数据计算模块的主要功能是将分布式文件系统中的日志信息转化为系统所需要的性能分析数据，性能数据计算模块使用Scope脚本与C#语言相结合的方式计算产品性能。

性能数据计算模块的主要流程为：导入数据、分离属性、属性重写、判断日志功能、计算相关数据、最后输出计算结果，具体的流程如图4-4所示。性能数据计算模块中每个节点的详细解释如下：

（1）导入数据，从Cosmos分布式文件系统中根据参数导入数据，Cosmos分布式文件系统中的数据包含客户端日志和服务器端日志，计算程序需要根据计算需求导入不同的数据；

（2）分离属性，指将日志中以字符串形式表示的属性数据分离出来。因为传输日志的过程中，客户端会将日志以字符串形式传输，为了方便处理日志数据，计算程序需要根据Key-Value的属性值将日志数据转化为字典格式再分别读取其中的属性值；

（3）属性重写，指对日志中保存的数据进行统一化、规范化处理。客户端需要获取用户信息，但是由于不同的手机厂商和不同的收集操作系统的不同API接口会使获取的信息结果不同，例如为了获取客户端的网络制式，一些操作系统会返回“WIFI”有的则会返回“Wi-Fi”，获取运营商时，中国移动可能会被返回“China Mobile”也有的会直接返回“中国移动”，属性重写这一步就是为了统一属性规范，使后面的计算更加方便；



图4-4 性能数据计算模块流程图

（4）判断日志的功能，日志的Event Name代表这条日志是产品哪个功能部分的日志，性能数据计算模块可以通过区分Event Name判断日志服务的功能模块，进而计算相关功能模块的性能数据；

（5）计算性能数据，这一步主要计算三种数据：功能模块的成功率、功能模块的延迟时间、每个功能模块的使用人数和每个功能模块当天的新用户量和次日用户存留率；

（6）将计算的三种不同的数据以不同的数据格式输出到文件中；在保存功能模块的成功率时，需要保存操作发生的时间、功能的名称、产品的版本号、网络情况、系统版本号、功能的使用次数，功能被成功使用的次数和功能的成功率；在保存功能模块的延迟时间时，需要保存操作发生的时间、功能的名称、产品的版本号、网络情况、系统版本号、功能的使用次数以及PLT75、PLT90和PLT95的值。在存储功能的使用人数时，需要保存操作发生的时间、功能模块的使用人数、功能模块的当天的新用户量和次日用户存留率。

表4-2 成功率数据表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 说明 | 是否可以为空 |
| Date | datetime | 当前数据产生的时间 | 不可为空 |
| AvailName | varchar（50） | 功能模块名称 | 不可为空 |
| DomainName | varchar（50） | 子功能模块名称 | 可以为空 |
| SuccessCount | int | 功能模块成功完成次数 | 不可为空 |
| RequestCount | int | 功能模块被请求次数 | 不可为空 |
| Avail | float | 当前功能模块的成功率 | 不可为空 |
| Network | varchar（50） | 客户端所在手机的网络类型 | 可以为空 |
| Market | varchar（10） | 发布产品的区域 | 不可为空 |
| OS | varchar（10） | 操作系统类型 | 不可为空 |

在性能数据计算模块进行设计的过程中，需要对存储数据的数据表进行设计。根据上一章的数据库设计，微软小娜性能分析系统需要如下几个数据表：功能模块成功率数据表（AvailCoA、AvailCoI）；按照网络分类后的功能模块成功率数据表，如表4-2所示；功能模块延迟时间数据表（LatencyCoA、LatencyCoI）；按照网络分类后的功能模块延迟时间数据表，如表4-3所示；功能模块用户量数据表，如表4-4所示。

表4-3 延迟时间数据表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 说明 | 是否可以为空 |
| Date | datetime | 当前数据产生的时间 | 不可为空 |
| LatencyName | varchar（50） | 功能模块名称 | 不可为空 |
| DomainName | varchar（50） | 子功能模块名称 | 可以为空 |
| RequestCount | int | 功能模块被请求次数 | 不可为空 |
| PLT75 | int | 75%用户使用模块的延迟时间 | 不可为空 |
| PLT90 | int | 90%用户使用模块的延迟时间 | 不可为空 |
| PLT95 | int | 95%用户使用模块的延迟时间 | 不可为空 |
| Network | varchar（50） | 客户端所在手机的网络类型 | 可以为空 |
| Market | varchar（10） | 发布产品的区域 | 不可为空 |
| OS | varchar（10） | 操作系统类型 | 不可为空 |

表4-4 用户量数据表结构

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 类型 | 说明 | 是否可以为空 |
| Date | datetime | 当前数据产生的时间 | 不可为空 |
| ScenarioName | varchar（50） | 功能模块名称 | 不可为空 |
| DomainName | varchar（50） | 子功能模块名称 | 可以为空 |
| UserCount | int | 功能模块活跃用户量 | 不可为空 |
| RequestCount | int | 功能模块被使用次数 | 不可为空 |
| Market | varchar（10） | 发布产品的区域 | 不可为空 |
| Network | varchar（50） | 客户端所在手机的类型 | 可以为空 |
| OS | varchar（10） | 操作系统类型 | 不可为空 |

1. 自动化工具模块设计

微软小娜性能分析系统需要获取每天用户的操作日志，进而分析小娜性能，但是如果每天人工提交脚本导入数据会占用很多时间。为了提高整个系统的自动化，本系统加入了自动提交脚本和自动导入数据的工具，其中定时运行程序采用的方法是Windows系统的计划任务，利用计划任务，可以将任何脚本、程序或文档定时运行。

为了使自动化工具经过一次部署后就可以一直运行，需要将参数写在XML文件中，当添加脚本删除脚本更改路径时，只需要更改XML文件而不需要重新部署，XML文件包括提交脚本和导入数据两个部分，提交数据部分主要结构有：脚本是否准备好，脚本名称，脚本所在路径，脚本优先级，日志数据路径等；导入数据部分主要结构有：是否需要导入，任务名称，源文件路径，数据格式文件，数据库名称，数据库表名等。

自动化工具模块主要包括：提交计算脚本任务、导入数据任务和异常情况处理任务。其功能设计图，如图4-5所示。

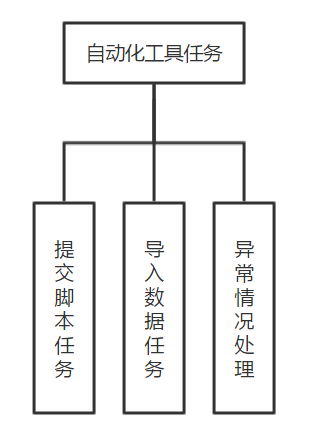


图4-5 自动化工具模块功能设计图

自动化工具模块用于日志收集模块之后（即已经将日志收集在Cosmos分布式文件系统中），计算数据模块之前，主要目的是将计算数据模块的计算脚本定时提交到Cosmos平台上并且将计算好的数据从Cosmos文件系统上下载下来并导入到数据库中。

在整个自动化工具中，提交脚本任务主要负责将计算产品性能数据的Scope脚本提交到Cosmos平台上运行，在这个过程中需要根据提交脚本的日志和业务需求获取需要计算性能数据的日期，并且需要从XML文件中获取提交脚本时必须的一些参数，包括：脚本路径、脚本优先级、脚本运行所占资源数和日志路径等。提交脚本任务时序图，如图4-6所示。其中，主程序先根据计算日志选择没有没计算的日期，并且根据业务需求得出需要计算的日期，然后主程序需要在XML文件中获取必须的参数，最后将计算日期、参数和脚本传递给脚本提交程序。



图4-6 提交脚本任务时序图

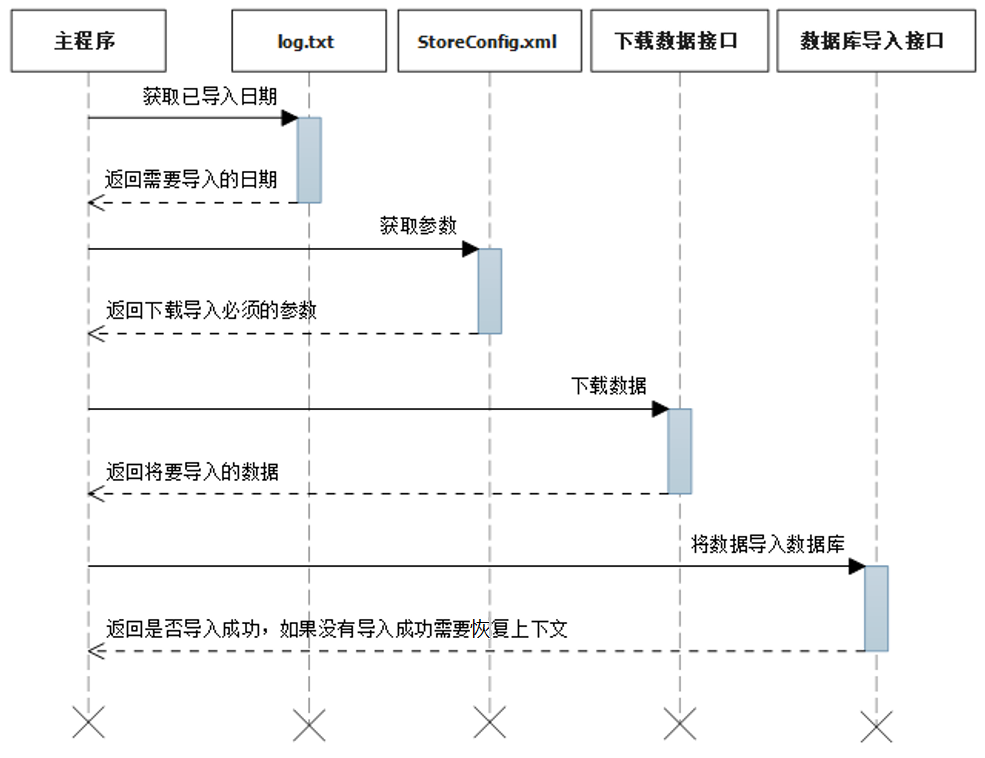


图4-7 导入数据任务时序图

导入数据任务负责将Cosmos分布式文件系统中的计算好的性能数据下载到本地服务器中，然后将这些性能数据导入到相应的数据库表中。在这个过程中，需要从XML文件中获取一些参数，包括需要下载的源文件在Cosmos文件系统中的路径和将下载数据保存在本地服务器的本地路径，获取性能数据文件名，获取文件格式类型，获取数据库名称和表名称。

导入数据任务时序图，如图4-7所示。在图中可以看到，主程序先根据导入数据的日志选择没有被导入的日期，并且根据业务需求得出需要导入数据的日期，然后主程序需要在XML文件中获取必须的参数，接下来主程序需要根据这些参数和日期在Cosmos分布式文件系统上下载需要导入的数据，最后将这些数据导入到相应的数据库表中。



图4-8 异常情况处理时序图

异常情况处理任务时序图，如图4-8所示。异常情况处理主要包括三个方面的异常情况：提交脚本是出现异常情况，下载数据时出现异常情况和导入数据库时出现异常情况。提交脚本时可能出现的异常情况有:脚本编译错误，计算时期所对应的日志源文件不存在，脚本计算过程中出现由于脚本逻辑问题出现的异常情况。下载数据时可能出现的异常情况有需要下载的数据不存在和因为权限问题所引起的不能下载。导入数据库时可能出现的异常情况有数据库连接不上，数据表格式与下载文件格式不相同，性能数据中出现个别异常数据。如果在数据导入数据库时发生异常，程序需要将已经导入的数据删除，保证数据库数据的准确性与可信赖性。除此之外在发生异常时还应该发送提醒邮件给开发人员，使其尽快修复问题。

1. 数据展示模块设计

微软小娜性能分析系统中，数据展示模块主要通过网站展示实现。网站展示数据的方式为：用户点击需要查看产品性能数据的参数值，例如用户可以选择网络数据（Wi-Fi/4G/3G/2G）、市场分类（Zh-CN/En-US/En-IN/Ja-JP）、产品所在操作系统（Android/iOS）等。在用户选好条件参数值后，网站会根据用户的需求展示相关的数据，例如网站可以通过表格的方式展示各个功能模块的成功率和延迟时间，网站也可以通过折线图的方式展示2个月以来的各项数据，可以直观的看出性能数据的变化趋势图，网站也可以以饼状图的方式展示数据，可以直观的看出部分与整体的关系。

当用户点击所需查看的数据后，网页会将用户的需求传递给控制器，然后控制器根据所需查看的数据选择合适的数据类型，在将数据库读取的数据返回给控制器，控制器对数据进行整理并按照合适的表示方式返回给页面，整个流程如图4-9所示。



图4-9 数据展示时序图

根据网站功能间的时序图可以清楚地看到整个模块的具体流程，当网站页面将用户需求传递给控制器时，控制器需要分析需求并将需求参数传递给数据模型接口，以获取相关的数据模型，控制器还需根据这些数据模型在数据库的相关表格中获取数据，然后控制器需要对数据进行处理，最后控制器将这些处理后的性能数据返回给网站页面，网站页面再将这些数据以可视化的图表形式展示出来。

1. 本章小结

本章介绍微软小娜分析系统的总体设计和模块设计，依照可靠性原则、安全性原则、易用性原则、先进性原则和可扩展性原则进行总体设计，梳理了微软小娜性能分析系统的业务逻辑，整理出性能分析系统的功能设计，并结合微软小娜性能分析系统的总体设计规划，合理划分系统的功能模块。本章对每个功能模块都给出了较为详细的时序图，并对功能模块的实现流程做了详细的描述，为后续微软小娜性能分析系统功能模块快速合理的实现打下坚实的基础。

# 微软小娜性能分析系统实现

通过对微软小娜性能分析系统的总体设计以及详细功能设计，为系统的最终实现奠定了良好的基础，本章主要介绍微软小娜性能分析系统的开发环境以及部署环境，并具体地描述系统各个功能模块的实现。

1. 系统平台的搭建

微软小娜性能分析系统开发采用面向对象与面向过程相结合的设计开发方法，针对每个子系统选择相应的开发方法，开发框架主要选择微软的.NET框架，网页设计采用ASP.NET框架[29]，脚本的开发采用微软为Cosmos开发的Scope语言。

微软小娜性能分析系统开发使用的操作系统为Windows 10，部署服务器系统为Windows Server 2012，系统的开发工具使用Microsoft Visual Studio 2015，Web服务器使用IIS，数据库管理系统为SQL Server。服务器操作系统遵循最小安装与最小服务原则，即安装稳定的操作系统补丁，禁用不必要的服务。

1. 日志收集模块实现

日志收集模块的主要功能为将微软小娜客户端产生的日志数据发送到Aria日志收集平台。本模块主要实现为将性能的分析需要的产品信息和用户操作信息封装为类，并使用性能分析需要的数据初始化类，然后将这些封装好的日志信息发送到服务器。

本模块的实现主要在客户端，在客户端我们实现一个日志类Log，主要功能为添加日志中的各种属性，包含的数据成员有日志产生时间、产品发布市场、产品版本号、日志编号、日志所代表功能名称、语言、客户端网络情况、客户端操作系统版本和一个可以添加属性的字典成员。

日志类还包含一个GetTimeStamp()函数负责获取日志产生时间，日志产生时间需要精确到毫秒，还有一个负责获取日志编号的函数GetImpressionID()，日志编号是用来区分不同会话（Session）的日志；客户端也包含一个发送日志的LogEventInit类，LogEventInit类用单例模式实现，主要功能为初始化日志发送端和发送日志，包含一个构造函数LogEventInit(string TenantToken)负责客户端发送日志的初始化，包含一个LogEventSend()函数负责发送日志，日志收集模块类图如图5-1所示：



图5-1 日志收集模块类图

在LogEventInit中包含一个Tenant Token变量，Tenant Token是专属Aria日志收集平台的一个字符串，主要目的是将日志数据与微软小娜产品相关联起来，因此在初始化LogEventInit时需要用Tenant Token初始化日志控制器，除此之外还需要将发送日志后的回调函数注册到日志控制器中，最后生成一个负责发送日志的对象。

在Log中包含一个Impression ID变量，可以理解为日志的编号，日志收集模块为每次会话使用GetImpressionID()产生一个UUID编号，使用通用唯一标识符可以保证标号的唯一性。在每次生成日志时，日志收集模块希望获取产生日志的具体时间，这是采用时间戳的表示方法记录当前的时间。

当用户进行每一项操作或微软小娜客户端发生一次状态变化时，日志记录类会记录操作的各项数据信息，然后将日志封装并保存起来，日志封装采用将日志的属性名和属性值以Key-Value的方式插入字典中。等到一个固定的时间或日志积累到一定数量后将所有日志统一发送到Aria平台，这样做的目的是因为如果每次生成日志之后就马上发送会使发送次数过多，即会是客户端浪费更多电量，也会使Aria服务器增加负担。在每次发送日志后，注册在日志控制器中的日志回调函数会返回日志是否发送成功的信号，如果日志没有发送成功，日志收集模块会重新发送该日志。

1. 性能数据计算模块实现

性能数据计算模块采用Scope脚本语言与C#相结合的方式实现，在Scope中可以调用C#的相关函数。

在实现计算脚本语言之前，性能数据计算模块应该先把Scope脚本需要的C#相关函数实现，C#实现的计算函数主要有:

（1）SafeDivide(double nume, double deno)函数，该函数负责安全的计算两个数的除法，安全体现在当分母为零时返回-1而不是异常退出；

（2）GetPropertyDictionary(Dictionary<string, Bonded<Extension>> data)函数，该函数负责将原始日志的数据转化为属性名与属性值相对应的字典。这是因为Aria平台在存储数据时采用的是Bonded<Extension>格式，计算模块首先需要将数据格式进行转化，这就要用到GetPropertyDictionary()函数；

（3）GetProperty(Dictionary<string, string> PropertyDict, string keyName)函数，该函数是将上一步获得的PropertyDict字典作为参数，然后选择与keyName相同的属性名返回其属性值，如果没有找到属性名相同的元素就返回null；

（4）RewriteNetwork(string provider, string typeName)函数，该函数的目的是将各个原始日志进行标准化处理，provider指网络运营商，typeName指网络制式，该函数的最终结果是所有网络运营商写作：中国移动、中国联通或中国电信，所有网络类型根据其网络制式与传输速度划分为：Wi-Fi网络、2G网络、3G网络和4G网络;

（5）RewriteDomain(string actionUrl)函数，该函数的目的是将每个actionUrl转换为功能名称，因为在获取的原始日志中，每个功能模块会有很多不同的actionUrl，例如闹钟功能会有打开和关闭闹钟，但是都属于闹钟功能模块。该函数将所有actionUrl进行标准化处理，将功能分类为网页搜索、闹钟、打电话、发短信、启动应用程序、寻找联系人、作笔记、提醒和日期；

（6）IsReleaseVersion(string version)函数，该函数的目的是判断一个版本号是否是正式发布版本，因为在开发测试过程中，一个功能模块的日志会出现在很多版本中，为了使性能分析的结构更加可信，数据计算模块需要对版本号进行判断，判断的基本标准为版本号必须以-release结尾，不能包含dev和rc，必须有4为版本号，且每一位以点号分隔。

脚本计算的第一步是将需要计算的日志从整个日志文件中取出，主要的几个条件为:开始日期、结束日期、版本号、微软小娜产品序列号和日志名称，根据这几个关键信息，性能数据计算模块将需要的日志取出。

脚本计算的第二步是规范化日志数据，使用GetPropertyDictionary()、RewriteNetwork()和RewriteDomain()三个函数规范日志数据。日志的计算主要可以分为3部分：每个功能模块称功率的计算、每个功能模块延迟时间的计算和每个功能模块活跃用户量的计算。

本文以语音识别模块为例介绍性能数据计算模块的实现过程。为了计算语音识别模块的成功率，程序需要获取语音识别模块开始和语音识别模块结束的日志数据。语音识别模块成功率的计算方法为语音识别结束模块日志除以数语音开始模块日志数与用户取消语音识别的日志数之差，即为语音识别模块的成功率。

首先，将日志名为“i audio start” 的日志取出，这些日志代表语音识别开始；然后，将日志名为“i audio complete”的日志取出，这些日志代表语音识别完成；最后，将日志名为“i audio cancel”的日志取出，这些日志代表用户手动取消语音识别。

接下来让语音识别开始的日志与用户手动取消语音识别的日志做左连接（left join），连接条件为两条日志的编号相同，取出为被用户手动取消的日志。然后分别计算出没有被用户取消语音识别的日志数量和成功完成语音识别模块的日志数量。

最后将这两份日志数量按照日期相同做内连接（inner join），并用没有被用户取消语音识别的日志数量除以成功完成语音识别模块的日志数量，即语音识别模块这一天的成功率。

因为计算语音识别模块成功率的时候已经获得的语音识别开始和结束日志，因此在计算延迟时间时，就直接使用这些日志计算语音识别的延迟时间。首先对这两份日志按照日志产生日期和日志编码相等做内连接，并计算出时间差。这样性能数据计算模块就获得了每个用户每次操作的延迟时间。

计算功能模块延迟时间的第一种方法是求将每个用户的操作延迟时间做平均，平均值为功能模块的延迟时间，但是直接求平均值会产生一些缺点，因为平均值与每一个数据都有关，其中任何用户延迟时间的变动都会相应引起平均数的变动，极易受到极端值的影响，因此性能数据计算模块没有采用求平均值的方法计算延迟时间。

为了避免个别离群点对数据的影响，性能数据计算模块采用求分位数的方法计算延迟时间。首先将每个用户的延迟时间按从小到大递增排序，性能数据计算模块会挑选某些数据点，以便把数据分布成为大小相等的连贯集，分位数就是取自数据分布的每隔一定点的连贯集合。性能数据计算模块会拿出所有延迟时间的第75个百分位数、第90个百分位数、第95个百分位数作为整个功能模块的延迟时间。假设第75个百分位数为2000ms，则其代表的含义为：75%的用户延迟小于2000ms。这样就计算出了语音识别模块这天的延迟时间：PLT75、PLT90和PLT95。

在性能数据计算模块需要计算的第三类数据是每个功能模块的使用人数，采用每个性能模块开始的日志计算。本文还是以语音识别模块为例，使用语音识别模块开始的日志计算，首先将这些日志按照用户编码进行去重，然后计算总量就是这一天使用语音识别模块的用户量。除此之外，性能数据计算模块还需要计算语音识别这一天的新用户数，这时需要将这一天所有的用户编码与之前用户编码进行比较，选出在之前没有出现过的用户编码，再计算用户量即为语音识别这一天的新用户量。

性能数据计算模块在将所有性能数据计算完之后需要做的就是将三种计算数据进行归类，然后保存在三份文档中，为了方便后面导入数据库，保存的数据结构与数据库表结构相同。

1. 自动化工具模块实现

微软小娜分析系统需要每天运行性能数据计算模块分析产品性能，如果每天需要开发人员手动提交运行脚本会导致开发效率降低，因此微软小娜性能分析系统采用自动提交和自动导入数据的方法提高系统的执行效率，其中定时运行程序是采用Windows系统的计划任务，利用计划任务，可以将性能数据计算模块定时运行。

自动化工具模块的实现流程为：

（1）定时或手动触发自动化工具运行；

（2）根据已经日志信息确定需要计算的日期，具体为判断运行日期前一周的计算日期，将未计算的数据重新计算；

（3）将XML中保存的脚本参数信息和需要计算的日期组合成为脚本提交命令，XML文件中保存的主要信息有：脚本名、脚本路径、脚本优先级、脚本依赖文件路径、脚本所需资源数和需要计算的日期，提交脚本的XML文件格式如下：

|  |
| --- |
| <Submit>  <Task>  <Ready> True </Ready>  <TaskName> ChannelDistribution </TaskName>  <Type> Submit </Type>  <IsRecord> True </IsRecord>  <ScriptName> ChannelDistribution </ScriptName>  <ScriptPath> D:\DailyReportSendRobot\</ScriptPath>  <Priority> 999 </Priority>  <CheckExistsPath> MergedLog\_%Y\_%m\_%d.cortana.ss </CheckExistsPath>  <Tokens> 80 </Tokens>  <Params>  <StartDate> </StartDate>  <EndDate> </EndDate>  </Params>  </Task>  </Submit> |

（4）新起一个进程执行脚本提交命令，并在脚本提交的日志文件中完成提交脚本日志信息；

（5）根据已经日志信息确定需要导入数据的日期，具体为判断运行日期前一周的导入数据日期，将未导入的数据重新导入到数据库；

（6）将XML中保存的数据格式信息、数据库信息、数据表信息和需要导入数据的日期组合成为数据导入命令，XML文件中保存的主要信息有：数据文件名、需要导入数据的日期、数据的原路径、数据格式文件、数据格式文件路径、数据库服务器名、数据库名、数据库表名以及需要的本地路径，提交脚本的XML文件格式如下：

|  |
| --- |
| <Import>  <Task>  <Ready> True </Ready>  <TaskName> ChannelDistribution </TaskName>  <Type> Import </Type>  <IsRecord> True </IsRecord>  <IsAotuRunBasedOnToday> True </IsAotuRunBasedOnToday>  <AccumulatedstartDate> </AccumulatedstartDate>  <IsNotDailyJob> false </IsNotDailyJob>  <StartDate> 2015-07-04 </StartDate>  <EndDate> 2015-07-27 </EndDate>  <FileSubPathOnCosmos> my/%Y/%m/%d/ </FileSubPathOnCosmos>  <UseFormatFile> False </UseFormatFile>  <FormatFilePath> </FormatFilePath>  <DataBaseName> UsageReport </DataBaseName>  <ServerName> </ServerName>  <UserName> </UserName>  <Password> </Password>  <ImportFileInfo>    <AccumulatedUU> AccumulatedUU\_%Y\_%m\_%d.tsv </AccumulatedUU>    <UsagePerDevice> DailyBasicInfo\_%Y\_%m\_%d.tsv </UsagePerDevice>    <UsagePerDevice> BasicInfo\_%Y\_%m\_%d.tsv </UsagePerDevice>    <DailyOverallUU> DailyUU\_%Y\_%m\_%d.tsv </DailyOverallUU>  </ImportFileInfo>  <LocalPath> D:\DailyReportSendRobot\ReportDataPath\ </LocalPath>  </Task>  </Import> |

（7）新起一个进程执行数据导入命令，并在数据导入日志文件中完成数据导入日志信息。在导入数据时，自动化使用SQL Server提供的批处理命令（BCP），BCP是SQL Server中负责导入导出数据的一个命令行工具，是基于DB-Library的能以并行的方式高效的导入导出大批量数据。在使用BCP导入数据之前性能分析系统需要生成数据表的格式文件，命令为：

bcp [Table] format nul –f table.fmt –c –T

则可以生成数据表的格式文件，之后使用导入数据的命令：

bcp [Table] in data.txt –T –f table.fmt

将数据导入数据库中；

（8）待所有脚本提交完成，所有数据导入完成，程序退出。

在自动化工具中还有一项特别重要的任务就是异常处理，异常处理主要分为两步：第一步是在异常发生时，立即抓住异常并恢复上下文；第二步是在本地日志文件中记录异常的详细信息，供开发人员查看；第三步是邮件通知开发人员，使其尽快查看并修复出现的问题。

自动化工具模块为每个任务在本地维护四个日志文件：Log.txt，Errorlist.txt，RecordSubmit.txt，RecordImport.txt。Log.txt存放脚本提交时间和脚本名称；Errorlist.txt存放脚本提交或者数据导入时出现的异常详细信息，当出现异常时查看Errorlist.txt文件可以查看异常信息寻找解决方法；RecordSubmit.txt存放已经提交脚本的日期，目的是防止重复提交脚本；RecordImport.txt存放已经导入数据库的数据日期，目的是防止重复导入数据。



图5-2 自动化工具模块类图

自动化工具模块采用面向对象设计方法实现，其类图如图5-2所示，在图中可以看到自动化工具实现了StoreXMLHelper类、ImportTask类、SubmitTask类、CopyAndImport类、StoreConstant类、CommandAreaLibrary类和StoreScopeHelper类，接下来详细介绍每个类的功能及其实现方法。

StoreXMLHelper类的功能为将XML中存放的提交脚本和导入数据的具体参数提取出来，包含两个方法：List<SubmitTask> GetSubmitListFromXML(string xmlPath)和List<SubmitTask> GetImportListFromXML(string xmlPath)。因为这两个方法基本实现相同，本文以GetSubmitListFromXML()方法为例介绍此类的实现。GetSubmitListFromXML()方法使用XmlDocument对象创建XML文档对象，然后将XML文件的路径信息传递给XmlDocument对象，这样XmlDocument对象就可以直接访问XML文件中的每一个节点。GetSubmitListFromXML()方法使用XmlDocument对象遍历XML的每一个节点，并将XML中保存的提交脚本和导入数据的具体参数提取出来存放进链表中返回，供调用者使用。

SubmitTask类为提交脚本所需要的参数封装起来的对象，主要包括的数据成员有：ready（脚本是否可以提交）、taskName（任务名称）、scriptName（脚本名称）、scriptPath（脚本路径）、priority（脚本执行优先级）、startDate（起始日期）、endDate（结束日期）、recordFileName（记录文件名称）、checkDataExistsPath（脚本依赖文件路径）。

ImportTask类将XML中导入数据所需要的参数封装起来，主要包括的数据成员有：ready（数据是否应该导入）、taskName（任务名称）、startDate（起始日期）、endDate（结束日期）、fileSubPathOnCosmos（需要导入的文件在Cosmos上的路径）、formatFilePath（导入数据格式文件路径）、serverName（数据库服务器名称）、dataBaseName（数据库名称）、localPath（数据被下载的本地路径）。

CopyAndImport类将从配置文件中导入数据所需要的各种参数封装起来的ImportTask类进行整理，主要包括的数据成员有：fileName（需要导入数据库的数据文件名称）、formatFileName（该文件的存储格式）、localPath（该文件的本地路径）、serverName（数据库服务器名称）、dataTableFullName（数据库表名）、formatFilePath（该文件中数据的格式文件路径）。

StoreConstant类保存自动化工具模块中所需的一些常量，包括xmlName（XML文件名称）、vc（Cosmos上运行脚本的虚拟集群名称）、dateToCheckBack（执行脚本前需要向前查找的日期）、scopeSDKDirectory（Scope SDK的路径）。

CommandAreaLibrary类包括自动化工具模块中所需的一些通用操作，包括执行命令函数Execute(string command, int seconds)、根据日志信息获取已经执行日期函数GetRecordDates(string FilePath)、写日志函数WriteLog(string logs, string logFilePath)、获取数据表中数据的格式文件名GetBCPFormatFileName(string DataBaseName)。

Execute()函数采用多进程方法执行命令，首先函数拼接字符串组成命令行然后使用cmd.exe执行执行命令。GetRecordDates()函数读取日志信息，将已经执行的日期输出。WriteLog函数将制定的日志信息输出到相应的文件中。GetBCPFormatFileName()根据数据表名称和数据表与格式文件名称的规则返回格式文件名称。

StoreScopeHelper类主要是获取提交脚本和导入数据的一些命令行信息，包括获取提交脚本所需的命令行函数GetSubmitString(SubmitTask t)、获取导入数据所需的命令行函数GetCopyFileFromCosmos(CopyAndImport c)、获取BCP命令函数GetBCPString(CopyAndImport c)和检查Cosmos分布式文件系统路径是否存在的函数CheckExistsOnCosmos(string fileName)。GetSubmitString()将SubmitTask对象的基本数据成员按照格式

scope.exe submit -i [ScriptPath] -vc [VCName]

拼接为提交脚本的命令。GetCopyFileFromCosmos()函数将CopyAndImport对象的基本数据成员按照格式

scope.exe copy [CosmosFilePath] [LocalPath] -overwrite

拼接为下载文件的命令。

最后一个主窗口类Form1会将之前所有的类与方法组合起来调用，具体实现过程如下:

第一步使用配置文件初始化StoreXMLHelper类和StoreScopeHelper类

第二步调用SubmitTask()函数，SubmitTask()函数将GetSubmitListFromXML()获得的提交脚本参数与StoreConstant中的参数结合构成可以被cmd.exe执行的提交脚本命令行，然后调用Execute()执行命令行并将执行数据写入日志文件中；

第三步调用ImportTask ()函数，ImportTask ()函数将GetImportListFromXML()获得的导入数据参数与StoreConstant中的参数构成CopyAndImport类，然后将CopyAndImport类的数据拼接成可以被cmd.exe执行的导入数据命令行，然后调用Execute()执行命令行将数据导入到数据库中并将执行数据写入日志文件中。

以上内容介绍了自动化工具模块实现过程中需要的类和数据结构，通过实现上文提到的所有类和其数据成员以及方法本文实现了自动化工具模块。

1. 数据展示模块实现

数据展示模块使用搭建网站，让网页展示数据的模式实现。网站的搭建使用ASP .NET框架，部署在IIS服务器上。网站主要实现的页面有发布指标页面（Release Metric）、通信顺序进程页面（CSP）、网络划分后的性能指标页面（Perf By Network）、核心数据页面（Core Metric）和产品质量得分页面（Quality Score Card）。

发布指标页面是在产品发布之间建立，目的是为产品的发布提供性能参数。网络划分后的性能指标页面是在产品发布之后为了查找产品性能与网络制式以及运营商之间的关系建立的页面。网络划分后的性能指标页面是为了更加详细的得出网络传输过程的性能指标。核心数据页面是展示微软小娜产品中用户最关系的核心功能的性能数据。产品质量得分页面是展示现在产品进度，产品每个功能模块计划表的详细数据。

网站实现的类图如图5-3所示，可以看到本系统将整个解决方案分为4个工程完成，包括CrossMetric工程、DataBase工程，Business工程和Model工程。CrossMetric工程主要负责网页前端的数据交互；DataBase工程主要负责数据库部分，包括连接数据库和运行相关SQL语句；Business工程主要负责前端与后台的交互，将CrossMetric工程需要查询的数据交给Business工程依赖DataBase工程实现；Model工程主要是在系统中需要的数据模型。

网站的整个框架采用MVC的设计模式，将数据与视图进行分离，可以容易的修改应用层的数据和业务规则，具有耦合度低，重用性高，有利于软件工程化管理等优点。

首先介绍Model工程，在Model工程中主要有两个类：ChartProperty类和SearchCondition类。这两个类是整个数据展示模块的数据模型，ChartProperty类将展示图形所需要的信息封装起来，主要包含的数据成员有xInterval（横坐标间隔）、isVisibleLegend（是否显示图例）、title（图表标题）、yFormat（纵坐标数字表示格式）、xMax（横坐标最大值）、xMin（横坐标最小值）、yMax（纵坐标最大值）、yMin（纵坐标最小值）等。

SearchCondition类封装了查找数据时需要的信息，主要包含的数据成员有date（日期）、OS（操作系统版本）、deviceType（设备类型）、market（微软小娜发布市场）、network（网络制式与网络运营商）、appVersion（微软小娜版本号）等。



图5-3 网站实现模块类图

接下来介绍DataBase工程，主要包括数据库的实现。DataBase工程含有两个类：SqlDBClient类和DBHelperProvider类。SQLDBClient类主要负责数据库的连接和SQL语句的执行，主要采用ADO.NET框架实现，使用了Connection、Command、DataAdapter、DataReader、DataSet等类进行数据库的连接与查询。SQLDBClient类的数据成员包含一个数据库连接对象connection，connection使用DbProviderFactory类的CreateConnection()方法初始化，并通过数据库连接串与数据库建立连接。

CreateConnection()方法实现代码如下。

|  |
| --- |
| public static DbConnection CreateConnection(string connectionString)  {  DbProviderFactory dbfactory =  DbProviderFactories.GetFactory(\_dbProviderName);  var connection = dbfactory.CreateConnection();  connection.ConnectionString = connectionString;  connection.Open();  return connection;  } |

SQLDBClient类还包含函数：ExecuteCommand(string safeSql)，GetDataSet(string safeSql)和GetFirstStringData(string safeSql)。

ExecuteCommand(string safeSql)方法用来执行insert、delete、update等SQL语句的,返回的是影响的行数，实现方法为使用connection对象的CreateCommand()方法建立一个DbCommand类，并将SQL语句传递给CommandText成员变量，最后使用DbCommand类的ExecuteNonQuery()方法执行SQL语句，代码如下。

|  |
| --- |
| public int ExecuteCommand(string safeSql)  {  using (DbCommand cmd = Connection.CreateCommand())  {  cmd.CommandText = safeSql;  int result = cmd.ExecuteNonQuery();  return result;  }  } |

GetDataSet()方法负责获取符合条件的数据表，返回值是一个DataSet对象，表示一个数据集，代码如下。

|  |
| --- |
| public DataTable GetDataSet(string safeSql)  {  DataSet ds = new DataSet();  using (DbCommand cmd = Connection.CreateCommand())  {  cmd.CommandText = safeSql;  DbProviderFactory dbfactory =  DbProviderFactories.GetFactory(\_dbProviderName);  DbDataAdapter da = dbfactory.CreateDataAdapter();  da.SelectCommand = cmd;  da.Fill(ds);  connection.Close();  connection.Dispose();  return ds.Tables[0];  }  } |

GetFirstStringData()方法返回SQL语句执行后的第一个元素，也是需要构建一个DbCommand类，然后使用DbCommand类的ExecuteScalar()方法获取SQL语句执行后的第一个元素。DBHelperProvider类主要负责获取数据库连接串和使用连接串构造SQLDBClient，该类包含一个SqlDBClient类的成员变量，成员变量使用数据库连接串初始化，数据库连接字符串以Key-Value的方式存储在Web.Config中，具体格式如下：

|  |
| --- |
| <appSettings>  <add key=" SqlServer " value = " server = Server; database = DB; uid = UID; pwd = PWD; Max Pool Size = 100; Min Pool Size=5; Connect Timeout = 0; "/>  <add key = " SqlServerCoA " value = " server = Server; database = DB; uid = UID; pwd = PWD; Max Pool Size = 100; Min Pool Size = 5; Connect Timeout = 0; "/>  <add key = " SqlServerEN " value = " server = Server; database = DB; uid = UID; pwd = PWD; Max Pool Size = 100; Min Pool Size = 5; Connect Timeout = 0; "/>  </appSettings> |

Business工程主要负责网页与数据库的交互，包括传递数据库连接串和SQL语句。Business工程主要包括两个类：CoACtl类和ModelBase类，CoACtl类继承ModelBase类，ModelBase类是CoACtl类的基类。ModelBase类包含一个SqlDBClient类的成员变量和一个参数为AppSetting配置文件关键字的构造函数，构造函数负责获取数据库连接串并将数据库连接串传递给SqlDBClient对象。CoACtl类继承ModelBase类，包含一个参数为数据库名称的构造函数，此构造函数会调用ModelBase类的构造函数，并将参数传递给ModelBase类，用来初始化SqlDBClient对象。

CoACtl类为控制器类，主要目的是实现SQL语句的执行，为页面展示和数据的中间层，负责为页面提供数据，包含很多执行SQL语句的方法，这些方法使用SearchCondition对象传递参数，并将SearchCondition对象拼接为需要执行的SQL语句。因为执行SQL语句的函数实现方法类似，现以GetAvailByNetwork()方法为例讲述，实现代码如下。

|  |
| --- |
| public DataTable GetAvailByNetwork(SearchCondition condition,  string availName, string network)  {  string sqlStr = string.Format(@"  select BeginTime, AvailName,  SuccessRate, SuccessCount,  RequestCount  from [dbo].[SingleAvailByNetwork]  where BeginTime='{0}'  and AvailName = '{3}'  and Market='{1}'  and NetworkDetail like '{4}'  and OS = '{2}'",  condition.BeginTime, condition.Market,  condition.OS, availName, network + "%");  DataTable dt = \_DBHelper.getDataSet(sqlStr);  return dt;  } |

GetAvailByNetwork()方法可以获取以网络划分的功能模块的成功率，实现过程为使用SearchConditon对象中的参数以及功能模块名称和网络制式拼接SQL语句，然后将SQL语句传递给SqlDBClient对象，并使用SqlDBClient对象的GetDataSet()方法执行SQL语句并返回结果。

表5-1 Core Metric表结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Daily | | PLT75 EN-US | | PLT75 ZH-CN | |
|  |  |  |  |
| Proactive | |  |  |  |  |
| Proactive（W3C\_Load） | |  |  |  |  |
| Cat 3a（Answer） | Text |  |  |  |  |
| Text（W3C\_Load） |  |  |  |  |
| Speech with SR |  |  |  |  |
| Speech w/o SR |  |  |  |  |
| Speech w/o SR with UPL |  |  |  |  |
| Cat 3b（Web Result） | Text |  |  |  |  |
| Text（W3C\_Load） |  |  |  |  |
| Speech with SR |  |  |  |  |
| Speech w/o SR |  |  |  |  |
| Speech w/o SR with UPL |  |  |  |  |
| Speech Perf | |  |  |  |  |

最后介绍CrossMetric工程，CrossMetric工程包括网站页面布局的设计和页面响应的处理。CrossMetric工程包括发布指标页面、通信顺序进程页面、网络划分后的性能指标页面、核心数据页面和产品质量得分页面，本文以核心数据页面为例介绍页面及其相应处理实现，核心数据页面模块的实现包括一个CoreMetric.aspx页面布局文件和一个CoreMetric.aspx.cs响应处理文件。CoreMetric.aspx负责页面中数据的填充，以及折线图的绘制，在页面构造使采用ASP.NET技术更加方便的操作页面布局，核心数据页面中包含每天的数据和每个月的平均数据，每天数据的表格如表5-1所示。CoreMetric.aspx.cs负责页面中数据的获取以及折线图的构造在填充数据时为了防止过多的条件判断使结构冗余的缺点，使用提前做好数据与表格映射的方式进行优化。CoreMetric类继承System.Web.UI.Page类，CoreMetric类包含一个字符串与单元格名称的映射字典，保存数据与表格映射，包含一个SetMapping ()方法填充字典对象。当加载页面时，CoreMetric类首先会调用Page\_Load()函数，负责初始化整个页面，Page\_Load()函数代码如下所示。

|  |
| --- |
| protected void Page\_Load(object sender, EventArgs e)  {  if (!IsPostBack)  {  AddClickAttributeOnTableCell();  MappingStringAndCellID();  MappingStringAndCellIDWeekly();  tbxStartDate.Text =  Convert.ToDateTime(GetMaxDay()).ToString("MM/dd/yyyy");  InitRecentlyVersion();  FillCoreMetric();  FillCoreMetricWeekly();  if (HttpContext.Current.Request.QueryString["Flag"] != null)  {  Route(HttpContext.Current.Request.QueryString["Date"],  HttpContext.Current.Request.QueryString["CoAVersion"],  HttpContext.Current.Request.QueryString["CoIVersion"]);  }  }  } |

在Page\_Load()函数中有一个IsPostBack变量，用来判断页面是否是第一次加载，当页面是第一次渲染或者是一个PostBack返回之后的加载，IsPostBack变量为真，否则IsPostBack变量为为假。Page\_Load()函数根据IsPostBack变量的值进行不同的数据处理和页面展示。Page\_Load()函数根据数据库中到现在为止最大的日期填写页面数据。Page\_Load()函数会根据默认的选项在数据库中读取相应的数据并填写在表格中。

CoreMetric类还包含一个BtnSearch\_Click()方法，负责用户点击搜索之后的响应操作，代码如下。

|  |
| --- |
| protected void BtnSearch\_Click(object sender, EventArgs e)  {  if (!string.IsNullOrEmpty(tbxStartDate.Text))  {  if (Convert.ToDateTime(tbxStartDate.Text) > Convert.ToDateTime(GetMaxDay()))  {  tbxStartDate.Text = Convert.ToDateTime(GetMaxDay()).ToString("MM/dd/yyyy");  }  SearchCondition condition = new SearchCondition();  condition.BeginTime =  (Convert.ToDateTime(tbxStartDate.Text)).ToString("yyyy-MM-dd");  MappingStringAndCellID();  MappingStringAndCellIDWeekly();  FillCoreMetric(condition);  FillCoreMetricWeekly(condition);  }  if (HttpContext.Current.Request.QueryString["Flag"] != null)  {  Route(HttpContext.Current.Request.QueryString["Date"],  HttpContext.Current.Request.QueryString["CoAVersion"],  HttpContext.Current.Request.QueryString["CoIVersion"]);  }  } |

当用户点击搜索按钮后，BtnSearch\_Click()方法会根据用户的选择条件在数据库中读取数据，具体实现为根据用户选择构造SearchCondition对象，然后将SearchCondition对象传递给CoACtl对象执行SQL语句，在此之前CoACtl对象应该被数据库连接串初始化过，然后使用CoACtl对象从数据库获取的数据按照数据与表格映射的映射关系填表，最后对填写数据的单元格填充颜色。

1. 本章小结

本章根据微软小娜性能分析系统的模块设计，论述了系统的开发环境以及运行部署平台，具体地描述系统了各个功能模块的实现过程，对各个功能模块都给出了较为细致的类图，而且对类图进行了详细的描述，完整的实现了微软小娜性能分析系统。

# 测试

采用合理适当的测试方法，能够全面细致地检测系统各项功能，发现隐藏的问题，有利于系统的改进，进而提高系统的性能。

1. 测试环境及方法

测试环境部署服务器均采用Windows Server 2012操作系统，Web服务器使用IIS，数据库使用SQL Server。

由于微软小娜性能分析系统主要是实现数据分析计算和展示功能，因此采用黑盒测试方法测试微软小娜性能分析系统运行情况，以及每个功能模块的运行情况。黑盒测试，也称功能测试，在微软小娜性能分析系统的功能规格设计的前提下，通过对各个功能模块的应用，来检测微软小娜性能分析系统是否满足最初的产品需求。理论上来说，黑盒测试需要列举所有可能发生的场景和测试用例才能保证检查出存在的问题，因此本文列举了尽可能多的测试用例来检测微软小娜性能分析系统。

除此之外还需要对微软小娜性能分析系统本身的性能进行测试，测试数据计算延迟时间，数据更新时间以及网页响应时间。

1. 功能测试

本系统重点进行功能性测试，在微软小娜性能分析系统的功能需求的前提下，通过对各个功能模块模拟真实的生产环境、模拟真实的用户场景，来检测微软小娜性能分析系统是否满足最初的产品需求。

针对微软小娜性能分析系统中，具体的业务场景进行测试。微软小娜性能分析系统包括日志手机模块、性能数据计算模块、自动化工具模块和数据展示模块，分别针对这四个功能模块进行测试。

1. 日志收集模块

对于日志收集模块的测试，主要测试该模块在微软小娜客户端发送用户操作日志能否实现日志的分布式收集和在Cosmos分布式文件系统中的存储，测试用例如表6-1所示。

表6-1 日志收集模块测试用例表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模块名称 | 日志收集模块 | 编号 |  | |
| 测试时间 |  | 测试级别 | 功能测试 | |
| 测试功能 | 日志收集和存储 | | | |
| 用例说明 | 正确完整的收集和存储日志 | | | |
| 前置条件 | 输入 | 预期结果 | 测试结果 | 失败原因 |
| Aria平台已配置完成 | 在微软小娜客户端发送用户操作日志 | 在Aria平台可以收到日志而且日志需要包括系统需要的完整数据 | 通过 |  |
| Cosmos分布式文件系统配置完成 | 手动将日志传到Aria日志收集平台 | 在Cosmos分布式文件系统中可以看到正确的日志数据 | 通过 |  |

1. 性能数据计算模块

对于性能数据计算模块的测试，主要测试该模块在日志存储于分布式文件系统中时能否正确计算微软小娜功能的成功率、延迟时间以及用户量，测试用例如表6-2所示。

表6-2 性能数据计算模块测试用例表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模块名称 | 性能数据计算模块 | 编号 |  | |
| 测试时间 |  | 测试级别 | 功能测试 | |
| 测试功能 | 性能数据计算 | | | |
| 用例说明 | 正确计算产品功能的成功率、延迟时间以及用户量 | | | |
| 前置条件 | 输入 | 预期结果 | 测试结果 | 失败原因 |
| 日志存储在分布式文件系统中 | 将实现的计算成功率脚本提交到虚拟集群上运行 | 将获得的模块成功率与日志数据进行分析对比，数据结果相同 | 通过 |  |
| 日志存储在分布式文件系统中 | 将实现的计算延迟时间脚本提交到虚拟机群上运行 | 将运行结果与日志数据进行分析对比，数据结果相同 | 通过 |  |
| 日志存储在分布式文件系统中 | 将实现的计算活跃用户量脚本提交到虚拟机群上运行 | 将计算的用户量与日志数据进行分析对比，数据结果相同 | 通过 |  |

1. 自动化工具模块

对于自动化工具模块的测试，主要测试该模块能否定时执行、正确提交脚本、正确导入数据和自动异常处理，测试用例如表6-3所示。

表6-3自动化工具模块测试用例表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模块名称 | 自动化工具模块 | 编号 |  | |
| 测试时间 |  | 测试级别 | 功能测试 | |
| 测试功能 | 自动化工具是否能够定时执行 | | | |
| 用例说明 | 定时执行、正确提交脚本、正确导入数据和自动异常处理 | | | |
| 前置条件 | 输入 | 预期结果 | 测试结果 | 失败原因 |
| 无 | 在系统计划任务设置定时执行 | 自动化工具在规定时间开始执行 | 通过 |  |
| XML文件中已有配置参数信息 | 使用自动化工具提交 | 自动化工具将脚本提交到虚拟集群上运行 | 通过 |  |
| XML文件中已有配置参数信息 | 将数据文件放在指定目录下，使用自动化工具导入 | 数据被导入到正确的数据表中，数据没有缺失 | 通过 |  |
| XML文件中已有边界条件 | 实际情况与条件不符，运行自动化工具 | 自动化工具在异常日志中记录异常信息且发邮件给指定可开发人员 | 通过 |  |

1. 数据展示模块

对于数据展示模块的测试，主要测试网站布局、网站读取数据库内容、网站展示数据，测试用例如表6-4所示。

表6-4 数据展示模块测试用例表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 模块名称 | 数据展示模块 | 编号 |  | |
| 测试时间 |  | 测试级别 | 功能测试 | |
| 测试功能 | 网站的数据展示 | | | |
| 用例说明 | 网站布局、网站读取数据库内容、网站以图表样式中展示数据 | | | |
| 前置条件 | 输入 | 预期结果 | 测试结果 | 失败原因 |
| 业务需求与布局格式对应关系已实现 | 按照业务需求，网站页面自动生成相关布局格式 | 布局与期望一致 | 通过 |  |
| 数据库连接串保存在配置文件中 | 根据配置文件数据库连接串读取数据 | 成功读取指定数据 | 通过 |  |
| 无 | 将数据以饼状图与折现图的样式展示 | 成功以饼状图与折线图样式展示数据 | 通过 |  |

2. 性能测试

为了测试微软小娜性能分析系统自身的性能，需要对系统模块分别进行性能测试。性能测试采用模拟用户操作的方法进行，主要测试日志的收集最大数量、数据计算延迟、数据更新延迟和网站响应时间。本次性能测试分为六个阶段，如图6-1所示。



图6-1 性能测试步骤

规划测试是指规划测试的指标和测试的方法，创建虚拟脚本的目的是通过脚本模拟批量用户操作，创建测试方案就是根据虚拟脚本创建具体的测试方法，接下来是运行创建的测试方案，然后系统检测测试结果，最后进行结果分析，判断是否满足产品需要。

具体的测试方法和测试结果如表6-5所示。

表6-5 微软小娜性能测试方法及测试结果

|  |  |  |  |
| --- | --- | --- | --- |
| 性能描述 | 测试方法 | 测试结果 | 结果是否通过 |
| 模块每秒可以收集日志的最大数量 | 在短时间内发送超大规模日志测试模块的性能 | 系统可支持平均每秒5百万次日志收集 | 通过 |
| 数据计算延迟 | 随机时间更新分布式文件系统中的日志，统计自动计算数据的延迟时间 | 数据计算延迟时间不超过3个小时 | 通过 |
| 数据更新延迟 | 随机时间改变性能数据，统计自动更新数据库的延迟时间 | 数据更新延迟时间不超过3个小时 | 通过 |
| 网站响应时间 | 户请求数据到网页展示出数据的时间间隔 | 网站响应时间在200毫秒以内 | 通过 |

1. 性能数据分析

为了测试微软小娜性能分析系统的计算准确性，需要对性能数据进行具体分析。本章将对微软小娜语义理解、语音唤醒、产品加载和微软账号登陆四个功能模块进行分析，选取了一天的性能数据如表6-6所示。

表6-6 微软小娜产品一天的性能数据

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能 | 成功率 | 延迟时间（ms） | | |
| PLT75 | PLT90 | PLT95 |
| 语义理解 | 95.6% | 2209 | 3200 | 4697 |
| 语音唤醒 | 99.32% | 47 | 616 | 1346 |
| 产品加载 | 99.52% | 953 | 1908 | 3410 |
| 微软账号登陆 | 88.02% | 129 | 229 | 421 |

数据表示含义:语义理解功能模块的成功率为95.6%，75%的用户延迟时间小于2209毫秒，90%用户的延迟时间小于3200毫秒，95%的用户延迟时间小于4697毫秒；语音唤醒功能模块的成功率为99.32%，75%的用户延迟时间小于47毫秒，90%用户的延迟时间小于616毫秒，95%的用户的延迟时间小于1346毫秒；微软小娜客户端加载的成功率为99.52%，75%的用户的产品加载延迟时间小于953毫秒，90%的用户的产品加载延迟时间小于1908毫秒，95%的用户的产品加载延迟时间小于3410毫秒；微软账户登陆模块的成功率为88.02%，75%的用户的登陆延迟时间小于129毫秒，90%的用户的登陆延迟时间小于229毫秒，95%的用户的登陆延迟时间小于421毫秒。

本文通过详细分析原始日志并与系统的计算结果进行对比，显示微软小娜性能分析系统的计算结果正确。

1. 本章小结

本章介绍了系统的测试环境以及测试方案，列举了系统的功能测试用例，描述了系统性能测试的方法并对性能数据进行了具体分析，测试结果良好，本系统的设计实现能够满足性能分析系统的实际需求。

# 总结与展望

1. 总结

本文首先介绍了微软小娜性能分析系统的研究背景及其研究意义，从性能分析系统的实际应用出发，梳理出性能分析系统的基本需求，确定了系统的建设目标，细化了系统的功能模块，并对系统进行总体设计和模块设计，实现了日志收集模块、性能数据计算模块、自动化工具模块和数据展示模块四大模块，最后对系统进行测试，保证了系统的正确稳定运行。

该系统已于2015年下半年上线，到目前为止系统已经过4轮迭代，并运行良好，对微软小娜产品提供可靠准确的性能数据。

使用微软小娜性能分析系统在微软小娜产品正式发布之前提供了量化的发布指标，为微软小娜的正式发布提供了性能上的数据支持。在开发过程中，分析系统通过分析产品数据，更快的发现产品中隐含的问题，帮助开发人员更快的定位和解决问题。在微软小娜产品发布之后，系统分析产品运行数据，分析用户数据，分析用户使用习惯，为下一个版本的产品提供了建设性的意见。

1. 展望

随着微软小娜产品的扩展，微软小娜的功能会越来越多，产品的更新也会越来越快，这势必会给微软小娜性能分析系统带来更多的扩展和挑战，微软小娜性能分析系统需要不断适应变化和挑战。

随着产品功能模块的数量不断增加和不断深入细化，如何在众多数据中提取真正有效的关键的数据，如何获取用户最真实的需求，如何判断合理、有效、快速定位产品的问题将是需要深入研究的问题。通过获取用户最真实的需求，快速定位产品问题，合理判断产品性能可以帮助技术人员和产品人员进行应用的改进和产品决策。

随着微软小娜产品的更新，性能分析系统可能需要多次迭代开发，以满足千变万化的业务需求。微软小娜性能分析系统在进一步开发中应该具有良好的开发和维护接口，这样可以迅速根据产品功能模块变化进行部署和再次开发。在对微软小娜性能分析系统进行迭代开发时，需要考虑功能模块变化、数据库变化、展示布局变化等多个层面的变化特性，需要对具体业务进行充分调研和分析，进行详细的设计和严谨的论证，以求具有对于需求变更的快速更改适应能力。

通过完善微软小娜性能分析系统，未来可以推广到其它的产品中，为其他产品提供性能分析数据，将数据处理和数据展示模块做成平台，可以灵活方便快捷的适应其他产品。

# 参考文献

1. 艾媒咨询. 2016上半年中国手机桌面行业研究报告[DB/OL]. http://www.iimedia.cn/43763.  
   html. [2016-08-01]
2. Hellerstein J L, Ma S, Perng C S. Discovering actionable patterns in event data[J]. IBM Systems Journal, 2002, 41(3): 475-493.
3. 张拥华．基于云平台的分布式日志系统设计与实现[J]. 信息与电脑（理论版）, 2015(17): 136-137
4. 廖湘科, 李姗姗, 董威. 2015. 大规模软件系统日志研究现状和进展[J/OL]. 软件学报，http://www.cnki.net/kcms/detail/11.2560.TP.20151109.1526.001.html
5. Xu W, Huang L, Fox A, et al. Detecting large-scale system problems by mining console logs[C]. Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles, 2009: 117-132.
6. Mariani L, Pastore F. Automated identification of failure causes in system logs[C]. Software Reliability Engineering, 2008: 117-126.
7. Xu W, Huang L, Fox A, et al. Mining Console Logs for Large-Scale System Problem Detection[J]. SysML, 2008(8): 4
8. 郎为民, 杨德鹏. 云计算中的分布式文件系统[J]. 电信快报, 2012(2): 7-10
9. Aimsam. Scribe[DB/OL]. https://github.com/facebookarchive/scribe/wiki. [2016-04-10]
10. 张川, 邓珍荣, 邓星, 黄文明. 基于Chukwa的大规模日志智能监测收集方法[J]. 计算机工程与设计, 2014(9): 291-297
11. 董新华, 李瑞轩, 周湾湾. Hadoop系统性能优化与功能增强综述[J]. 计算机研究与发展, 2013(S2): 8-22
12. 张晓龙. 基于Flume的XML数据自动收集系统[J]. 科学技术与工程, 2013(30): 206-210
13. 亢丽芸, 王效岳, 白如江. MapReduce原理及其主要实现平台分析[J]. 现代图书情报技术, 2012(2): 65-72
14. Richter J. CLR via c#[M]. Redmond: Microsoft Press, 2006.
15. Jamie T. Get to Know SQL Server 2012's SQL Server Data Tools[J]. SQL Server Pro, 2013, Vol.15 (5)
16. 闫旭. 浅谈SQL Server数据库的特点和基本功能[J]. 价值工程, 2012(22): 239-241
17. Jayleen H. SQL Server 2014 Cumulative Update 3 Now Available[J]. SQL Server Pro, 2014
18. Han J, Pei J, Kamber M. Data mining: concepts and techniques[M]. Elsevier, 2011.
19. Dino E. Which ASP.NET Is Better? [J]. InformationWeek, 2011(1307): 50-52
20. 李园, 陈世平. MVC设计模式在ASP.NET平台中的应用[J]. 计算机工程与设计, 2009(13): 146-150
21. Anderson T. Delve into ASP.Net MVC[J]. Personal computer world. 2009, 32(7)
22. 林勤花. 基于.NET的MVC + LINQ框架的应用[J]. 信息技术, 2013(7) : 167-168．
23. Gupta P, Govil M C. MVC Design Pattern for the multi framework distributed applications using XML, spring and struts framework[J]. International Journal on Computer Science and Engineering, 2010, Vol.2 (4): 1047
24. 高宏, 王婷婷. 基于MVC和Entity Framework的团餐系统[J]. 计算机与现代化, 2014(12): 68-72
25. Kennedy J. What's a Universal Windows Platform (UWP) app? [DB/OL] MSDN. https://msdn.microsoft.com/en-us/windows/uwp/get-started/whats-a-uwp. [2016-04-12]
26. Whitney T. Guide to Universal Windows Platform (UWP) apps. [DB/OL] MSDN. https://msdn.microsoft.com/en-us/windows/uwp/get-started/universal-application-platform-guide. [2016-04-12]
27. 江凌, 杨平利, 杨梅, 袁媛. 基于ADO.NET技术访问SQL Server数据库的编程实现[J]. 计算机应用技术, 2014(08)
28. 中华人民共和国国家标准. GB/T 17969.8-2010. 通用唯一标识符(UUID)的生成和登记及其用作ASN.1客体标识符部件. 中国电子技术标准化研究所, 2011
29. 仰燕兰, 金晓雪, 叶桦. ASP.NET AJAX框架研究及其在Web开发中的应用[J]. 计算机应用与软件, 2011(06): 201-204
30. 杨锋英, 刘会超. 基于Hadoop的在线网络日志分析系统研究[J]. 计算机应用与软件, 2014(08): 317-322
31. Silberschatz A, Korth H F, Sudarshan S. Database system concepts[M]. New York: McGraw-Hill, 1997.
32. 李锦川, 钱秀槟, 方星. 基于国产操作系统的网络日志管理系统构建[J]. 计算机安全, 2010 (10): 59-61．
33. 周琪锋. 基于网络日志的安全审计系统的研究与设计[J]. 计算机技术与发展, 2009, 19(11): 139-142, 146.
34. 胡光民, 周亮, 柯立新. 基于 Hadoop 的网络日志分析系统研究[J]. 电脑知识与技术, 2010, 6(22): 6163-6164.
35. Yuan D, Park S, Huang P, et al. Be Conservative: Enhancing Failure Diagnosis with Proactive Logging[C]. Proceedings of the 10th Symposium on Operating Systems Design and Implementation (OSDI), 2012: 293-306.
36. Yuan D, Mai H, Xiong W, et al. SherLog: error diagnosis by connecting clues from run-time logs[C]. ACM SIGARCH Computer Architecture News, 2010, 38(1): 143-154.

# 致谢

转眼间研究生生涯即将结束，在论文即将完成之际，在此感谢在研究生期间对我的学习和生活给予帮助的所有人。

论文的主要工作是我在微软实习期间完成，在实习期间我的企业导师Ethan和微软小娜的整个团队都给了我极大的帮助，小娜团队严谨而专注的工作态度深深影响着我，在此衷心感谢企业导师和微软小娜团队对我的关心与指导。

郑启龙老师细心指导我完成了论文的写作工作，从论文宣统、论文开题、论文初稿到论文的最终完成给予了我细心的指导和不懈的支持。郑启龙老师对我的论文提出了许多宝贵的意见，在此表示衷心的感谢。

感谢父母，感谢家人，他们无尽的支持和无私的关爱使我能够在学校专心完成学业。感谢在读研期间认识的小伙伴，他们丰富了我的业余生活，他们给我的学习和生活带来了很多快乐和帮助。

感谢中科大软件学院的各位专业课老师对我两年半来的传授解惑，感谢教学部学工部等学院老师对我的帮助。

特别感谢在百忙之中评审论文的各位专家老师！

论文修改说明

论文中存在的问题：

（1） 自动获取日志是在线还是离线？

修改：在模块设计部分（4.2.1节）增加介绍日志获取的方式（在线还是离线）。

（2） 实现部分更多的是介绍要实现类的说明，具体实现技术少，补充一些关键类方法实现的具体内容。

修改：实现部分（5.5节）增加介绍类方法的具体内容并辅以代码解释。

（3）目录等应符合学校论文规范，如页码大小、格式应统一。

修改：页码字体改为Times New Roman，目录中Abstract字体改为Times New Roman。

（4）系统测试内容应加强，测试结果可截取实际测试中的相关界面，来证明系统测试的准确性。

修改：加强测试部分（6.2节和6.3节），针对模块进行功能测试，增加性能测试方法步骤。由于已经离职所以未能截取测试界面。

（5）微软小娜是微软发布的全球第一款个人智能助理，作者没有说清楚微软小娜与该论文关系，在功能与性能方面有什么区别和不同？

修改：在摘要中增加介绍微软小娜背景介绍，在1.1节中介绍微软小娜功能和性能。

（6）文章的结构不紧凑，第三章需求分析是为第四章系统结构设计准备的，是其一部分，第五章系统功能模块是第四章系统结构设计的进一步解释，建议三章合并。

修改：合并第四章总体设计和第五章结构设计。

（7）第四章系统结构设计中不能体现系统性设计，建议增加系统结构设计图描述之。

修改：在第四章增加系统结构设计图（图4-1）。