GSM话务网管数据采集系统的设计与实现

计算机应用技术专业

研究生 孔珏　指导老师 李志蜀

摘　要：随着电信网络规模的增长，电信业务的扩充，电信运营者对网络管理的需求也不断增加、改动。计算机产业的飞速发展无疑为以高效、智能为目标的电信网网络管理技术提供了根本的保证。本文所述的省级 GSM 话务网管系统已经经历了一期、二期工程，并在移动、联通共计 54 个省级项目，陆续开展三期工程的开发和施工。作者进入与四川大学计算机学院合作的亿阳信通公司实习，加入 GSM 话务网管三期项目，从事网管底层数据采集平台的开发工作。作者在职期间负责全网项目中诺基亚和摩托罗拉两厂商所有通信设备的性能和资源数据采集开发。目前，GSM 网项目已全面投入应用，运行稳定。

数据采集平台是网管系统的数据接入层，它负责从各厂商设备采集原始数据，进行归一化处理，为网管系统所有中上层应用提供统一的数据平台，因此数据采集系统是网管建设非常核心以及重要的部分。

论文具体组织如下：

第一章，概述移动通信网管系统的产生历史。

第二章，阐述了电信网络管理的理论背景。

第三章，详述 GSM 网性能数据采集系统的设计，并分析如何

既实现高效、准确的数据采集，又保持系统应用的灵活性、可扩展性；在采集系统的分析过程中，本文将穿插介绍相应的技术原理及特色，并且与二期系统进行一定对比。

第四章，举一个由作者开发的具体设备采集实例，揭示通信设备数据采集的实现过程。

第五章，文章简略概述系统正在着手进行的软件升级和功能扩展情况。

第六章，结束语。

关键词：电信网络管理系统、数据采集、LDAP、文件数据库

The Design and Implementation of Data Collection in GSM network management system

Computer Applications

Graduate KONG Jue **Advisor** LI Zhishu

Abstract： With the growth of telecommunication network, the requirement of network management is increasing.And the high-speed development of computer science guarantees the high-efficiency and intelligence of telecommunication network management system. The GSM network management in this paper ,which is developed by BOCO Inter-Telecom Corporation cooperated with Computer Department, Sichuan University, has already been through one period, second period, and now is going to be third period in 54 provinces of CMCC and UNICOM . I have been engaged in the development of data collection system in the third period project, and all the equipments of Motorola and Nokia is my task. Now, the GSM network management system has already been applied and is running stably.

The data collection flat is the data-accessed layer of network management system. The flat collects the data from many kinds of

telecommunication equipments which are made by different manufacturers, and finally presents a uniform data flat to the other applications of GSM network management system.

This paper is organized as follows:

Chapter 1, the history of mobile telecommunication network management system.

Chapter 2, I describe the background theory of telecommunication network management system.

Chapter 3, the paper analyzes the GSM network data collection system. At the same time, I give the characters of technology in third period project which are better than the second.

Chapter 4, I give an instance to describe the implement of data collection which is both high-efficient and flexible.

Chapter 5, analyzing problems and talking about the trend of our system development.

Chapter 6, the summarization.

KeyWords: Telecommunication Management Network System, Data Collection, LDAP, Text Database

1 引言

中国通信市场不断扩大，电信网络规模的增长速度惊人，每个电信网络的运营者都趋向拥有可以经营多种业务和可以负担大量用户的电信网络。电信网络的规模在增长，电信网络内的设备也在增长，不同类型的设备，不同厂商的设备出现在同一个电信网络内，而不同类型的设备有不同的特性，需要不同的管理方式，不同厂商的设备有不同的管理接口，提供不同的管理信息，并且这些管理接口、管理信息随时可能变动。如何管理这样庞大复杂的电信网络是电信运营者所面对的问题，也是电信网网络管理所面临的一大问题。

GSM 话务网是目前移动通信的主体运营网络，网络规模大、结构复杂、业务总类多。电信运营商为了发展的需要，在总部和省一级移动电话网络管理中心建设综合网络管理系统，以实现集中化管理与维护。本文介绍的省级 GSM 话务网网管系统，是对移动通信网络管理的全面、整体以及可持续发展的综合解决方案。网管上层应用按照运营商的需求向其提供有效的管理数据，而这些以多种形式呈现的数据均源自网管底层的数据采集平台。要实现不同厂商、不同设备数据快速、准确采集入库，就需要一个健壮、高效，同时又具备可扩展性的数据采集系统。可以说，数据采集平台在网管系统中扮演了一个核心并且重要的角色，它屏蔽了底层网络设备的异构问题，为网管中上层应用提供了统一的数据平台。

2 背景知识

2．1 TMN基本原理

TMN（Telecommunication Management Network，电信管理网络）是ITU－T（国际电信联盟）提出来的关于网络管理系统化的解决方案。在CCITT的推荐标准M.3010中对TMN作了如下的定义，即 TMN 提供一个有组织的体系结构，以实现各种类型的运营系统和（或）电信设备的互联，同时使用一种共同的结构和包括协议和消息的标准化的接口来交换管理信息。电信管理网将一系列网络管理功能综合起来，通过采集网络工作参数、控制相应设备等，协调整个网络的正常运营。

TMN 和电信网的关系如下图 2.1 所示：

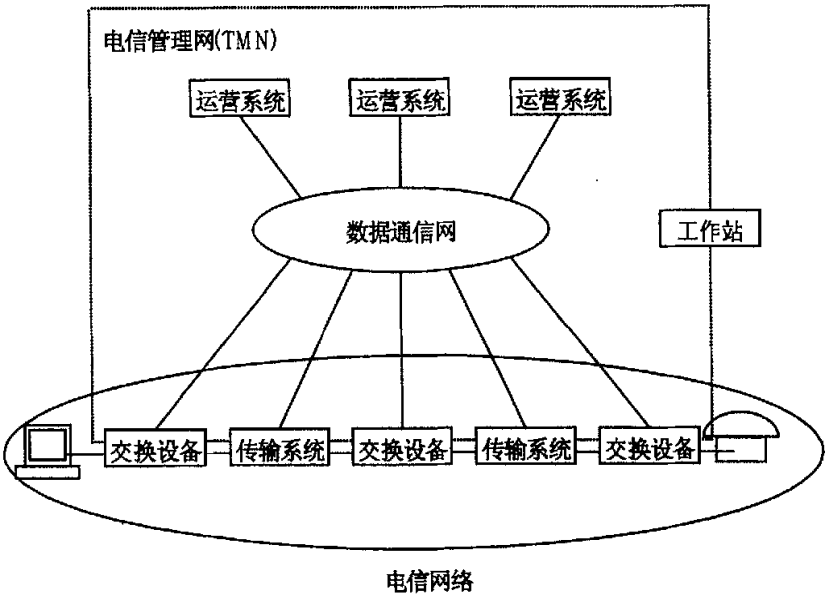


图 2.1 TMN 与电信网关系图

2．2 GSM数字移动通信原理

GSM (Global System for Mobile Communication，全球移动通信系统) 是最重要的第二代移动通信标准之一，本文介绍的采集系统正是以采集GSM网络单元（简称网元）的数据为例。

GSM系统典型结构如下图2.2所示，可以看作四个子系统或功能实体组成，即MS、BSS、NSS、OSS。

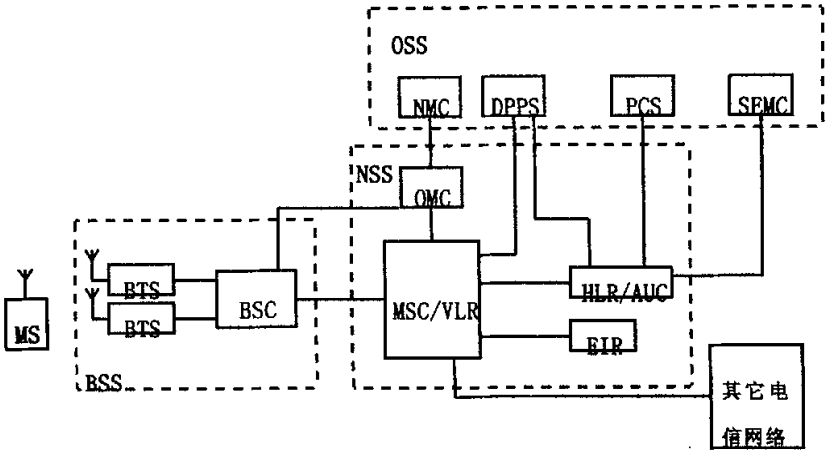


图2.2 GSM系统结构图

图示说明：

MS：Mobile Station,移动台

BSS：Base Station System，基站子系统

BSC：Base Station Controller，基站控制器

BTS：Base Transceiver Station，基站收发信台

NSS：Network Switching System，网络交换子系统

OSS：Operation Support System.，操作支持子系统

OMC：Operation and Maintenance Center，操作维护中心

MSC：Mobile-service Switching Center，移动业务交换中心

VLR：Visitor Location Register，来访用户位置寄存器

HLR：Home Locate Register，归属用户位置寄存器

AUC：Authentication Center，鉴权中心

EIR：Equipment Identification Register,移动设备识别寄存器

主要子系统和功能实体介绍如下：

（1）MS（移动台）

MS 是公用 GSM 移动通信网中用户使用的设备。

（2）BSS（基站子系统）

BSS 是由 BTS（基站收发信台）和 BSC（基站控制器）这两部分的功能实体组成，是 GSM 系统中与无线蜂窝方面关系最直接的基本组成部分。

（3）NSS（网络交换子系统）

NSS 由六个功能单元组成，即 MSC（移动业务交换中心）、HLR（归属用户位置寄存器）、VLR (访问用户位置寄存器）、AUC（鉴权中心）、EIR (移动设备识别寄存器) 、OMC（操作与维护中心）。 NSS 主要包含有 GSM 系统的交换功能和用于用户数据与移动性管理、安全性管理所需的数据库功能。

MSC 是网络的核心，它提供交换功能并面向系统其它功能实体， MSC 可从三种数据库，即 HLR、VLR 和 AUC 获取处理用户位置登记和呼叫请求所需的全部数据。反之，MSC 也根据其最新获取的信息请求更新数据库的部分数据。

OMC 用于对 GSM 系统的集中操作与维护，允许远程集中操作维护管理。OMC 根据与电信网络中所接设备类别不同分为管理无线设备的 OMC-R（OMC-Radio）和管理交换设备的 OMC-S（OMC-Switch)， OMC-R 对 BSS 进行操作和维护, OMC-S 对 NSS 进行操作和维护。网管系统对 GSM 网无线和交换两类设备数据采集，一般情况下通过管理无线和交换设备的 OMC 进行，有时候也通过设备直联网管系统进行。

（4）OSS（操作支持系统）

OSS 是相对独立的对 GSM 系统提供管理和服务功能的单元，包括移动用户管理、移动设备管理以及网路操作与控制。

2．3　省级 GSM 话务网管体系架构

三期 GSM 网络管理系统的体系架构主要分为 5 个层次，从底层业务到上层应用依次为：数据采集层，数据处理层，核心服务层，应用服务层，应用展现层。软件结构如下图 2.3：

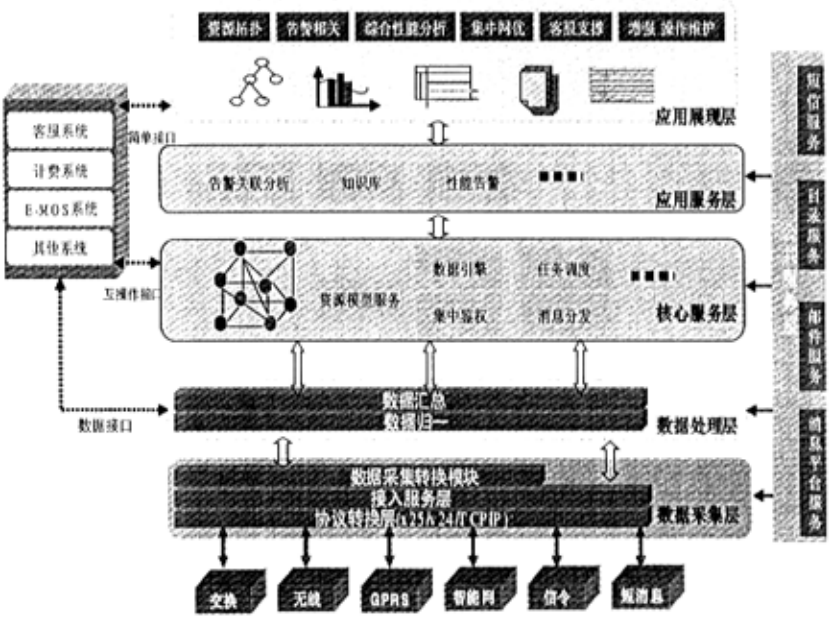


图 2.3　网管软件体系结构图

数据采集层主要完成接入协议转换、接入服务和数据采集的功能，完成网络的配置、性能、告警数据采集和网元操作维护的接口管理。

数据处理层完成数据在时间、地域、网元等各种维度上的数据汇总工作，为上层应用提供不同维度和粒度的预处理数据。

核心服务层完成对象的标准化建模工作，并提供系统关键服务，提高模块重用程度，便于系统统一管理和集中控制。

应用服务层完成系统核心业务，为上层程序提供基于业务的操作管理功能。

应用展现层为用户直接提供基于 C/S、B/S 架构的 GUI(图形用户界面）功能。

2．4　网管数据采集系统

采集平台是网管系统的数据接入层，它负责从不同厂商的通信设备采集原始数据，进行算法映射，形成归一化数据。网元数据采集类型有两种：配置管理数据和性能管理数据。为完成这两种数据采集任务，项目组设计了两个相对独立的采集系统：NRM 配置采集系统和 NPM 性能采集系统。NRM（Network Resource Management）采集网元的配置管理数据，NPM（Network Performance Management）采集网元的性能管理数据。

本文将仅对 NPM 性能数据采集系统进行分析，而不再介绍 NRM 采集系统，主要基于以下原因：

（1） 性能数据采集时间频繁，数据量大，采集形式复杂多样。性能数据采集一般需要小时粒度，及时性要求高；而配置数据比较稳定，采集可以按情况一天一次，或几天一次。

（2） NPM 系统不仅包含了 NRM 系统的关键技术，并且软件体系设计更健壮，能够支持性能数据采集的复杂性和实时性。

（3） NRM 系统和 NPM 采集系统是两个互不影响的系统，可以独立运行。

3 NPM 系统设计分析

3．1 NPM 系统结构要解决的问题

二期网管系统在移动通信网络管理与维护中发挥了重要的作用，但随着对业务处理能力的要求越来越高，也出现了一定的局限性：

（1） 采集时间不灵活

数据采集程序以 UNIX 系统 Crontab 方式定时运行，人工设定不同设备采集时间及其间隔。要判断不同厂商设备数据的产生时间，需要长期的维护经验，因此系统对维护人员有很强的依赖型，而且无法适应数据产生时间发生突变的情况。

（2） 所采数据的完整性和及时性难以协调

一方面，运营商希望网元/OMC 端产生原始数据后，网管系统能够及时采集呈现，这需要采集时间尽量早；另一方面，网元的数据产生时间有一定的波动范围，为了在执行采集程序时确保网元数据已经产生，需要将程序执行时间定的适当晚，才能保证数据的完整性。要人为确定一个最佳时间非常困难，并且还无法预防设备出现突发事件所导致的数据延迟数个小时的情况。

（3） 网元配置信息管理有待加强

二期系统对所有网元配置信息诸如主机地址，接口信息等都通过一个 INI 配置文件完成，并定时备份此文件。但是一个简单的 INI 文件不能配置更复杂的网元信息，传统的文件备份方式也存在很大安全隐患。

（4） 网管核心数据库负担过大

二期系统中，网管核心数据库同时负责数据采集以及其它中上层任务。采集系统对核心数据库频繁的进行大规模数据操作，增加、修改、删除大量临时表及临时数据，数据库服务器负担过重，并且已经开始影响到网管中上层应用。

3．2 NPM 采集系统结构

三期网管工程的性能数据采集系统我们称之为 NPM 采集系统，它由采集配置管理、采集控制、采集实现三大部分组成。

系统架构如图 3.1 所示：

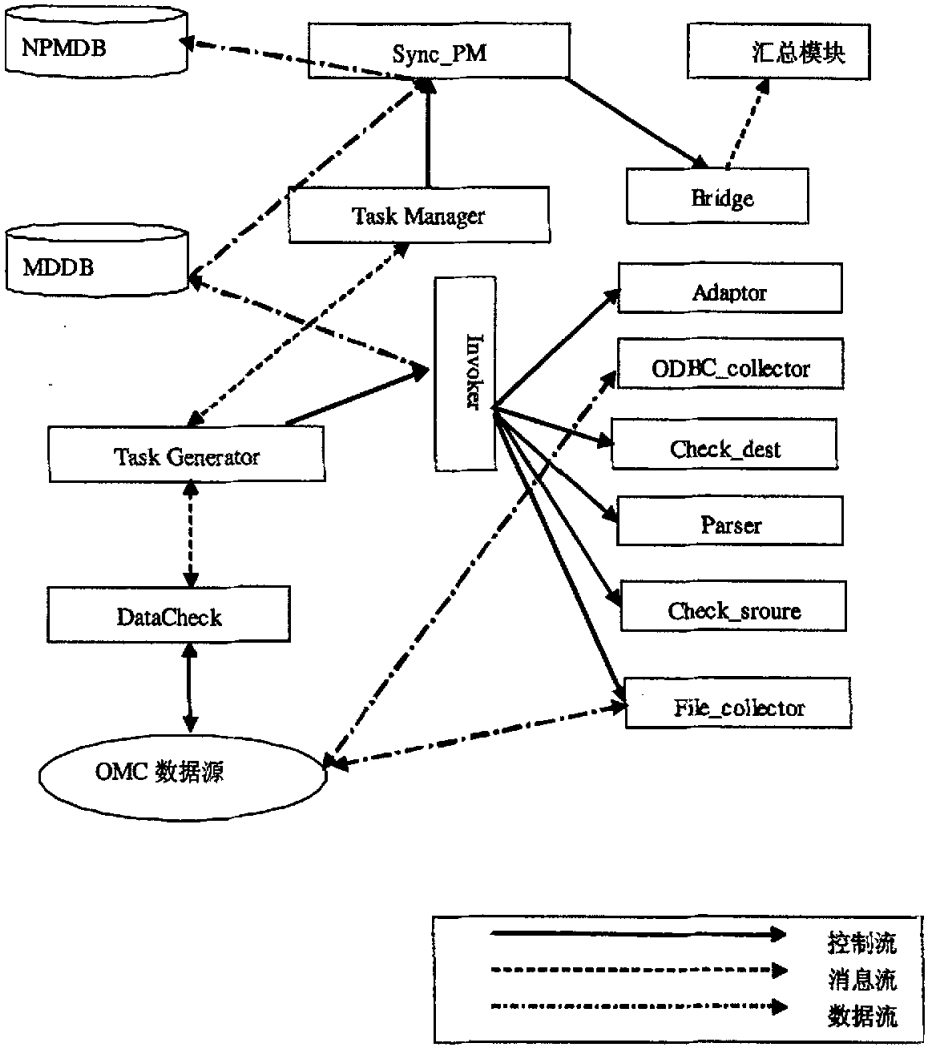


图 3.1 NPM 性能采集系统架构

NPM：网管性能管理平台，它将负责 GSM 移动话务网络性能数据的采集、存储、分析，并且是网管中上层应用以及其它支撑系统获得性能数据的唯一渠道。

NPMDB：网管性能管理平台核心数据库，包括网管系统的底层、中层和上层应用数据。

MD：网管数据采集服务器

MDDB：采集服务器数据库，它有限期的存储数据采集过程中使用的临时表和临时数据。三期系统采用独立的数据库辅助采集过程，减轻核心数据库 NPMDB 的压力，而 NPMDB 所使用的底层表数据由 MDDB 同步得到。

OMC 数据源：通常设备的无线和交换数据都来自于管理无线和交换设备的 OMC，也有部分数据采自直联网元。

汇总模块：是网管性能管理平台中层模块。

系统通过数据源检测模块（DataCheck）按照一定的时间和网元粒度来检查数据的完整性，这种检测需根据不同厂商设备特性完成，基本上是定时（可以配置）去轮询数据源的方式。一旦数据的生成情况符合设置的完整性要求的时候，就发“采集任务请求”消息给采集任务生成器（Task_Generator），让它负责去调度实际采集任务的执行并进行状态跟踪，采集任务由 Invoker 负责协调采集过程的各步骤，将设备性能数据采集到 MDDB 中。任务管理器 Task_Manager 主要用来协调多采集服务器的数据同步请求，调度同步任务，将采集到 MDDB 中的数据同步到 NPMDB 中，并由 Sync_PM 调用 Bridge 将汇总任务消息发至汇总模块，通知执行网管中层数据处理。

3．3 采集配置管理部分

网管二期系统采用一个 INI 文件存储所有厂家 OMC/网元的配置信息。

比如一个标志为 OSS303 的 OMC 部分配置信息存储格式如下：

```
[OSS303]
OMC_HOST        =$OMC_IP        #OMC 主机 IP 地址
OMC_PWD         =$OMC_PWD       #OMC 登陆密码
OMC_UID         =$OMC_UID       #OMC 登陆用户
VENDORNAME      =MOTOROLA       #OMC 所属厂商
VERSION         =1620           #设备版本
…… …… ……
```

二期系统的配置管理相对于三期系统，有以下几个不足点：

第一，旧系统的采集配置管理只是一个平面的一层结构，难以表示一个具有多层结构的模型。

第二，二期系统中不同程序模块间为共享配置数据产生了很多临时文件，在使用 LDAP 技术以后，数据共享不再主要使用临时文件方式，而是通过访问 LDAP 服务器获得，系统性能得到优化。

第三，二期系统的配置文件采用定时进行文件备份的维护方式，存在比较大的安全隐患。

三期系统所有厂商设备以及系统自身有关的配置信息都存放在 LDAP 服务器上，通过使用 LDAP 技术，不同应用有了统一的操作接口。

3．3．1 LDAP 技术原理

LDAP（Lightweight Directory Access Protocol 轻型目录访问协议）是一种部分基于 X.500 目录标准的开放标准，但更简单、更精练且可扩展性更好。LDAP 目录是一种数据库，但它不是关系型数据库，而是一个树状的目录型数据库，信息被集中存储在 LDAP 服务器上的 LDAP 目录中。

LDAP 目录结构与 UNIX 文件系统非常相似，数据按层次存储。基本组成元素有 DN、OU、数据项。“根”或“基本 DN”（Distinguished Name，专有名称），类似于 UNIX 系统的根目录；目录被进一步细分成组织单元 OU（Organization Units）；有一些 OU 下一级仍旧是 OU，类似于 UNIX 系统中文件夹下的文件夹，而

有些 OU 中包含数据项，这是整个树的叶子部分，这种“树一叶”结构使 LDAP 变得易于扩展。LDAP 技术的主要特点有：

（1） 目录型数据库和关系型数据库最基本的不同是，目录服务器是为执行快速查询而设计的，而关系型数据库还需要为事务处理等复杂应用进行优化。

（2） 目录是一个典型的层次结构，可以进行特定的数据管理。

（3） LDAP 专门制定了一套标准，处理数据访问，数据库通信等问题， LDAP 应用程序接口可以隐藏协议的细节问题。

（4）LDAP 无需专门的 DBA 管理数据库，服务器易于安装。

通过使用 LDAP 协议，客户机将查询发送给 LDAP 服务器（从技术上讲，LDAP 没有“读”功能；客户机通过将搜索请求发送给服务器来“读”目录项）。服务器检查客户机权限（即，客户机有权访问数据库吗？可以读被请求的树吗？可以将信息写入数据库吗？可以删除项吗？，等等），然后返回请求信息。几乎所有的现代编程语言都有 LDAP 的 API，这意味着几乎任何一个软件都可以支持 LDAP。LDAP 的应用系统见下图 3.2：

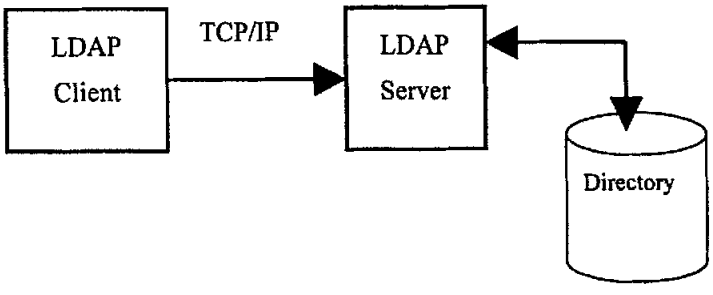


图 3.2 LDAP 应用系统结构

3．3．2 LDAP_API 模块设计

采集系统采用 Perl 作为编程语言，Perl 模块 Net:: LDAP 可以用于搜索 LDAP 目录，增加、删除、修改目录或数据项信息。采集系统以 Net::LDAP 为基础编写 LDAP_API（LDAP_API.pm）模块，提供对 LDAP 服务器的访问接口，模块实现的主要功能如下表 3.1：

名称	接口参数	说明
new		对象初始化
get_attr_by_Filter	ldap_base：搜索起点 ldap_filter：搜索目标	按照关键字法搜索，通过 Hash 表获取搜索节点的各项属性
get_attr_by_DSN	ldap_base：搜索起点 ldap_filter：搜索目标	通过准确的 DSN 名称搜索使用 Hash 表获取搜索节点的各项属性
add_entry	entry_name：目录名 ldap_base：搜索起点 attr_hash（Hash 表）：增加的属性	增加指定节点及其属性
delete_entry	ldap_base：搜索起点 entry_name：节点名	删除指定节点
add_attr	ldap_base：搜索起点 entry_name：节点名 attr_has（Hash 表）：增加的属性	增加指定节点某属性
delete_attr	ldap_base：搜索起点 entry_name：节点名 attr_array（Hash 表）：待删除属性	删除指定节点某属性
modify_attr	ldap_base：搜索起点 entry_name：节点名 attr_array（数组）：待删除属性	修改指定节点某属性

表 3.1 LDAP_API 模块

3．3．3 LDAP 树结构

NPM 采集系统的 LDAP 树结构示例如下图 3.3：

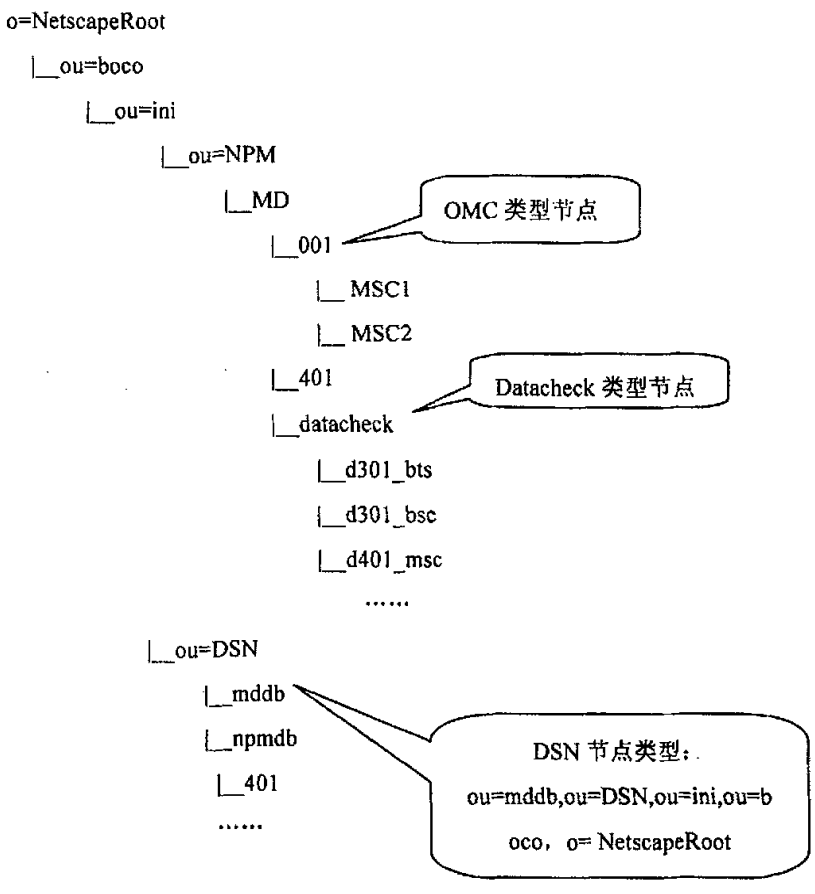


图 3.3 LDAP 树结构示意图

o=NetscapeRoot 是 LDAP 目录的根,采集系统所有的配置信息都属于组织单元 ou=boco 以下层次。节点的专有名称 DN 由根到此节点中的一系列 DN 组成，它们之间用逗号间隔。比如 DN 是“ou=301,ou=DSN,ou=ini,ou=boco，o= NetscapeRoot”的节点，表达了如下的层次关系：

o=NetscapeRoot

|__ou=boco

|__ou=ini

|__ou=DSN

|__ ou=301

3．3．4　采集系统 LDAP 节点类型

（1） OMC 类型节点：列出 OMC 的详细配置信息。

属性	值例	描述
objectclass	omc	节点类型
pm_asm_list	TPM_MSC_NOKIA|TPM_RADIO_BTS_NOKIA|TPD_CELL_HO_NOKIA|TPD_RADIO_BSC_NOKIA|TPM_TKGP_NOKIA|TPM_DESTCODE_NOKIA	采集的目标以短竖线间隔，所列举参数在专门的映射文件中有定义
oss_txt_dsn	401_txt	数据源
oss_db_dsn	401	数据源
version	T12	OMC 版本
vendor_id	4	设备厂商数字标识
pm_common_mapping	/opt/BOCO.DAL/NPM/mapping/nokia/release/pub_pm_nokia_2003.map	算法映射文件（即 Mapping 文件）路径
omc_user	User	Omc user
omc_pwd	Password	Omc passwd
dsn_list	401	All dsn list for datacheck
vendor_name	nokia	
BASE_NODE	ou=MD,ou=NPM,ou=ini,ou=boco,o=NetScapeRoot	节点路径
OU_NAME	401	OU 名称

（2）DSN 类型节点：列出某 OMC 的数据源信息。

属性	值例	描述
objectclass	dsn	节点类型
db_name	401	设备数据库名
db_user	user	DB user
db_pwd	passwd	DB passwd
db_type	odbc	数据库驱动类型
BASE_NODE	ou=DSN,ou=ini,ou=boco,o=NetScapeRoot	节点路径
OU_NAME	401	OU 名称

（3）Datacheck 类型节点：配置设备数据检测信息。

属性	值	描述
objectclass	datacheck	节点类型
omc_id	401	
check_id	d401_bts	进行 datacheck 操作的数据类型
restarat_cmd	DataCheck.pl -i d401_bts	Datacheck 启动命令
config_file	nokia_bts.xml	Datachek 的配置文件
invoker_cmd:	GP-NPM_invoker.pl	执行 Invoker 的命令格式
logfile	/opt/BOCO.DAL/NPM/trace/datacheck/d401_bts.log	日志路径
BASE_NODE	ou=datacheck,ou=MD,ou=NPM,ou=ini,ou=boco,o=NetScapeRoot	节点路径
OU_NAME	401	OU 名称

3．4　采集控制部分运行机制

3．4．1　采集控制机制

NPM 采集系统通过检测数据源生成情况，来控制采集任务并驱动日常采集。采集控制部分负责底层采集任务的生成调度，最终实现主动采集并及时通知汇总。

采集控制的基本机制如图 3.4 所示：

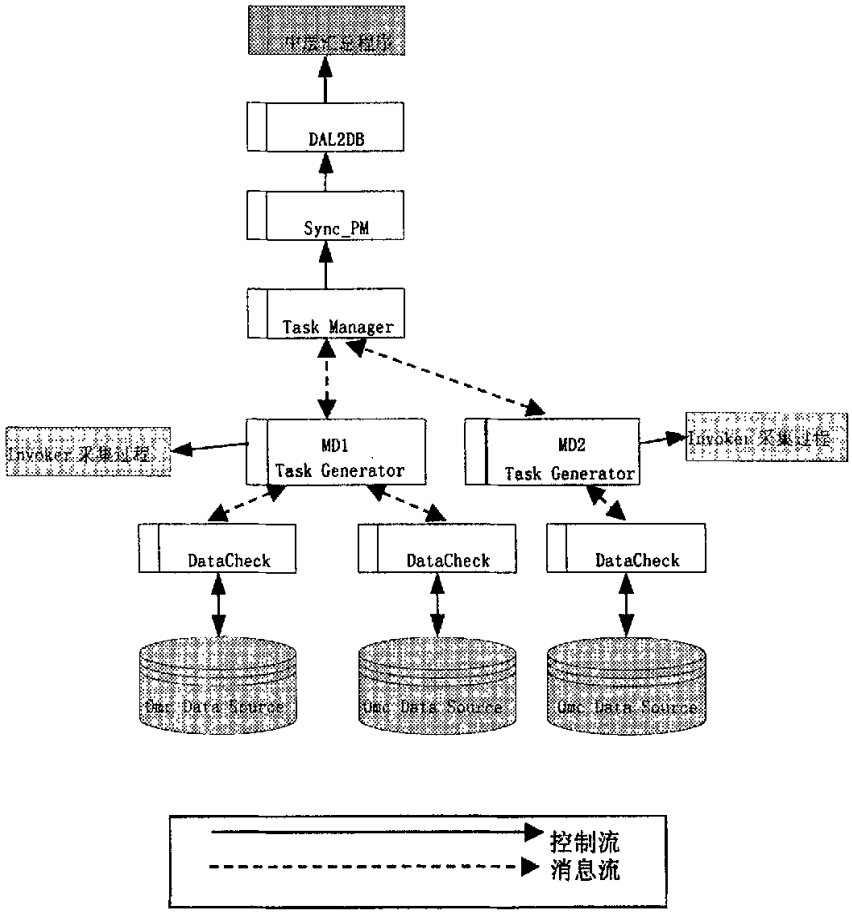


图 3.4 采集控制机制示意图

首先通过 DataCheck （数据源检测工具）来检测采集是否可行，然后生成采集任务发给特定 MD（采集服务器）上的 Task_Generator（采集任务生成器）,通过 Task_Generator 来调度 Invoker 采集过程；调度完成后，Task_Generator 向 Task_Manager（任务管理器）发送性能数据同步请求，Task_Manager 将协调多采集服务器的数据同步请求。Sync_PM 完成 MDDB 到 NPMDB 的数据同步后，向即时汇总调度器发送即时汇总请求。Dal2Db 负责管理从多个采集服务器（MD）发送的性能数据同步请求，协调即时汇总过程。

3．4．2 DataCheck 数据源检测工具

DataCheck 做为主动采集的发起方，是驻留采集服务器或 OMC 上的守护进程，它定时的对 OMC 的数据进行完整性检查，一旦某时间段数据符合采集条件，就向 Task_Generator 任务生成器发送采集任务消息，触发采集。

3．4．2．1 采集模式分类

对于不同设备，其数据产生与存储形式也相应不同，按设备采集接口类型分有数据库接口和文件接口两种模式。

数据库模式采集使用应用程序接口访问设备数据库。数据源检测原理通常是：检查特定表集合的数据是否有→根据比例判断是否已经符合采集条件→以上都符合，那么 DataCheck 将会认为一个特定的采集任务已经形成。

文件模式采集形式多样，有的设备定时产生数据到特定目录下，有的需要在设备主机发命令取文件，有的通过代理协议访问等等，待采数据可能存在于一个或几个文件中。文件模式检查数据的原理通常是：检查特定的文件列表中的文件是否存在→检查文件是否还在变化→判断文件大小→以上都符合，那么 DataCheck 将会认为一个特定的采集任务已经形成。

3．4．2．2 Datacheck 数据源检测工具

Datacheck 采用“核心程序+厂家模块”的方式，如图 3.5 所示：

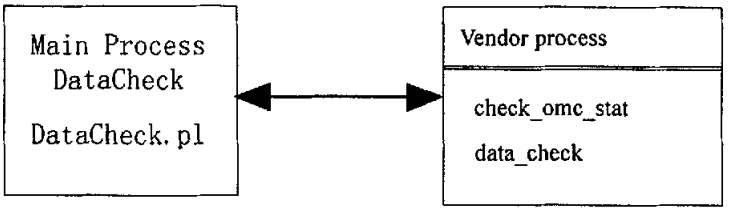


图 3.5 DataCheck 模块设计图

（1） 核心程序 DataCheck.pl

调用接口：DataCheck.pl –i $check_id

核心程序 DataCheck.pl 负责与厂家无关的流程处理，参数 $check_id 是 LDAP 目录中 Datacheck 类型节点的 OU 名，以此调用不同的数据源检测模块。

（2） 厂家模块

厂家模块由两部分组成：XML 配置文件和程序包。

XML 文件用于配置不同设备采集条件；程序包必须包括两个函数 check_omc_stat 和 data_check，check_omc_stat 检查厂家的网元/OMC 状态，data_check 检测待采集数据的准备情况。

3．4．3 Task_Generator 采集任务生成器

采集任务生成器是位于采集服务器上的守护进程，每台采集上均会运行一个 Task_Generator，负责从各个厂家的 DataCheck 模块接收消息，形成采集调度的实际任务。

任务生成器运行时会 fork 三个守护进程：

进程一：定时检查自己的三个后台进程是否还在运行，如果不正常，将状态记入日志。

进程二：以 socket 方式从 DataCheck 接收消息，并向 Task_Manager 发送消息。

- DataCheck 发送签到消息；
- 心跳检测消息：控制 DataCheck 的运行情况。如果某一个 DataCheck 心跳超时则重新启动它；
- 采集任务到达消息；
- 一个采集任务完成后向 Task Manager 发送性能同步、即时汇总消息。

进程三：接收消息队列的消息并进行处理，启动采集任务。

3．4．4 Task_Manager 任务管理器

任务管理器是运行在某一台采集服务器上的守护进程，负责管理从多个采集服务器（MD）发送的性能数据同步请求，协调调用性能同步（Sync_PM）过程，并负责监控采集服务器上的 Task_Generator 的运行情况。

任务管理器运行时会 fork 三个守护进程：

进程一：定时检查自己的三个后台进程是否还在运行，否则启动。定时查看表 dal_datacheck（存放 DataCheck 的运行信息）中的 last_updtime 字段，来判断该 MD 上的 Task_Generator 是否正常运行，否则 telnet 到相应 MD 上重启该 MD 上的 Task_Generator 程序

进程二：Socket 侦听 server,接收 Task_Generator 发送的 socket 消息，并进行相应处理。

- 签到消息：如果是首次签到，将在表 dal_datacheck 中插入一条记录；
- 心跳检测消息：均会把当时的时间 update 到表 dal_datacheck 中 last_updtime，以便第一个进程判断状态；
- 性能同步请求：则将该请求发送到消息队列中；

进程三：接收消息队列的消息并进行处理，启动同步程序 Sync_PM。

3．4．5　Sync_PM 性能同步

采集服务器MD将数据采集入MDDB的底层性能表，然后性能同步程序负责生成导出 MDDB 底层性能数据的脚本，并运行之生成数据文件，以 FTP 的方式传送到 NPM，再将文件导入 NPMDB 对应表。

3．5　采集实现部分运行机制

采集实现运行机制如图 3.6、图 3.7 所示：

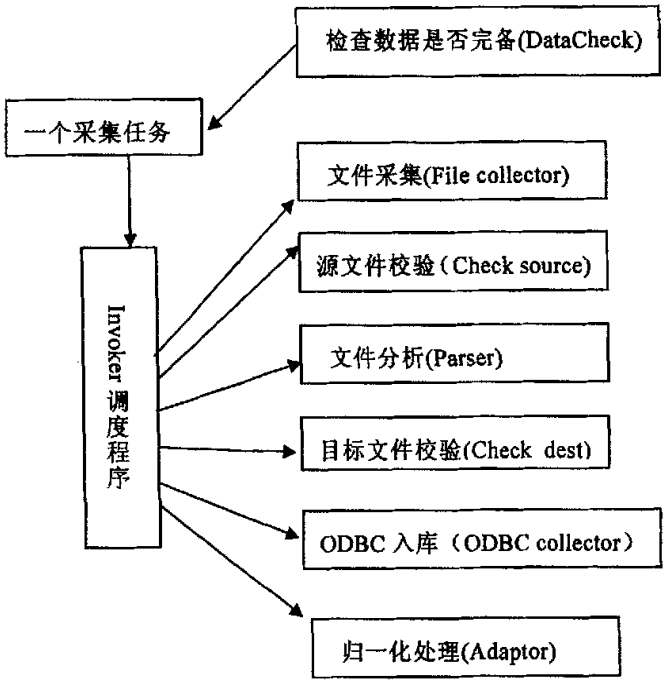


图 3.6 文件模式采集实现机制

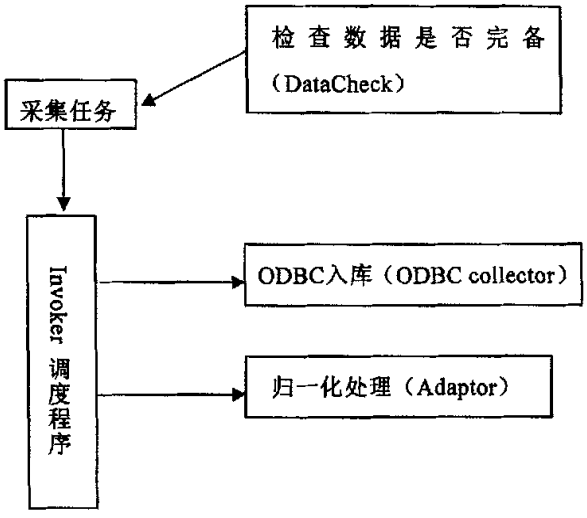


图 3.7 数据库模式采集实现机制

Invoker 是采集过程的总调度程序；File_collector 负责取得文件，Check_source 负责检查源文件中的错误并记录，Parser 负责解析源文件生成格式化的目的文件，Check_dest 负责检查目的文件的格式类型错误，ODBC_collector 负责将数据入 MDDB 临时表，Adaptor 将临时表数据整理入最终底层性能表。

3．5．1 Invoker

3．5．1．1 Invoker 设计

Invoker 负责全程调用采集的各个过程，工作流程如下图 3.8 所示：

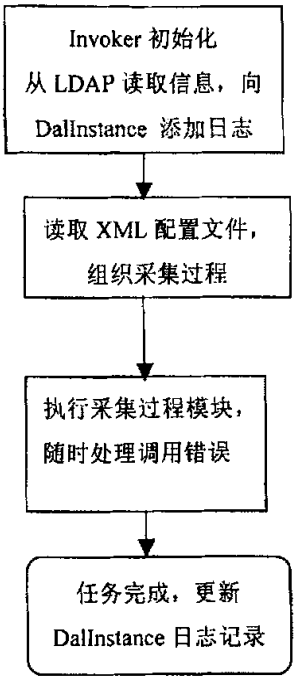


图 3.8 Invoker 流程图

Invoker 模块主程序仅仅相当于一个 XML 配置文件解释程序，采集的复杂流程是在一个名称为 GP-NPM_invoker.xml 的配置文件中定义，日志管理使用 DalInstance 设计。这样既为不同通信设备提供独立的采集模型设计，又将多样化的采集流程纳入统一的管理和日志记录。

3. 5. 1. 2 DalInstance

DalInstance 由两部分组成：

（1） Dal_Instance 表,记录性能采集实例；

（2） Perl 模块 DalInstance.pm，作为应用程序接口，Invoker

调用其访问 Dal_Instance 表。

设计 DalInstance 是为了将 Invoker 调度的各模块更紧密地联系在一起，便于跟踪，并且可以向其它应用提供数据，用于监控和任务管理。NPM 采集系统中，每一项性能采集都有一个唯一任务号，这个任务号由 Task_Generator 管理器生成，它贯穿于所有的 Invoker 调度的模块中，这样我们就可以通过任务号在 DalInstance 的数据库表中检索本次任务产生的各个实例，再通过他们的状态标志和错误号，我们就可以对失败的任务或实例进行处理。

3. 5. 1. 3 Invoker 配置文件 GP-NPM_invoker.xml

通过配置文件 GP-NPM_invoker.xml，Invoker 可以灵活的调用采集过程中的各个模块，并对这些模块出现的错误进行灵活的处理，实现完整的采集任务。配置文件的根元素是<invoker>,根节点下面有两个子节点<task_def>、<action_def>。<task_def>节为不同采集类型作出采集步骤定义，<action_def>具体定义采集步骤出现的不同行为模块，有关配置文件详细的参数解释将在后文的实例介绍中举例阐述。

3. 5. 1. 4 XML 解析器选择

XML 解析器是程序访问 XML 文档和结构的应用程序接口，有两种基本类型：基于树型的解析器和基于事件的解析器。

基于树型的解析器，将 XML 文档转换成树型结构。这类解析器分析整篇文章，同时提供一个 API 来访问所产生树的每个元素。其通用的标准为 DOM（Document Object Model，文档对象模型）。

基于事件的解析器，将 XML 文档视为一系列的事件。当一个特殊事件发生时，解析器将调用开发者提供的函数来处理。这些解析器从头到尾处理文档，并将类似于元素的开始、元素的结尾、特征数据的开始等等事件通过回调（callback）函数报告给应用程序。其通用标准为 SAX （Simple API for XML ）。

Expat 解析器， 是一个基于事件驱动（SAX）的非验证性解析

器。验证型解析器会检验 XML 格式的正确性，并根据文档类型定义（DTD）检验 XML 文件的有效性。而非验证型的解析器，只检测 XML 文件格式是否正确，具有非常高的速度和效率。

采集系统所使用的 XML 文件，没有文档类型定义（DTD），也没有关于 XML 的更多复杂应用，仅仅采用了 XML 的结构做配置文件。采集系统的这一特色非常适合采用 Expat 标准的基于事件驱动（SAX）的非验证性解析器。

3．5．1．5 Invoker 调用接口

Invoker 采集实现过程是在 Task_Generator 采集任务生成器中进行调用并记录日志，其调用接口基本形式为：

GP-NPM_invoker.pl -o <OSS Label> -task_id <Task_Id> -s <start_time> -e <Stop_time>

[-ne_type 采集的网元类型]
[-task_name 任务名,缺省是 PM 任务]
[-ne_list 要采集的网元列表]
[-meas_list 要采集的测量列表]
[-table_list 要采集的目标表列表]
[-log 日志开关]
[-step 执行采集的步骤列表，对应<step>的 action_name 属性]

其中“-o”与“-task_id”是必要的参数，分别为在 LDAP 中定义的 OMC 标识和由 Task_Generator 产生的唯一的任务 ID 号，它贯穿于 Invoker 调度的同一采集任务的所有模块中。“-s”和“-e”分别是采集数据的开始时间和结束时间。其余参数为可选参数。对于 Invoker 自己来说，只有参数“-o”，“-task_id”和“-step”是 Invoker 主程序 GP-NPM_invoker.pl 使用的，其他参数通过配置文件中定义的关系传给 Invoker 调用的程序。

3．5．2 ODBC_collector

3. 5. 2. 1 ODBC_collector 设计

ODBC_collector 模块根据配置文件中定义的源数据列表，通过 ODBC 接口或其他特定方式，将数据源中的数据在目标数据源中创建相应临时表保存，以便后续模块处理。

整个模块由一个主程序 GP-NPM_odbccollector.pl 和若干算法映射文件组成。

工程流程是：

（1） 通过 LDAP 协议获取全局变量和配置信息，得到所要采集的源数据信息；

（2） ODBC_collector 填写 dal_Instance 表，记录采集过程，日志文件；

（3） GP-NPM_odbccollector.pl 主程序通过读取 Mapping 算法映射文件，分析文件格式与内容，组合 SQL 语句；

（4） 根据已经生成的 SQL 语句，完成数据从源数据库表到 MDDB 临时表的过程。

程序使用到的重要函数模块有 DBIs 模块（DBIs.pm）和 DalInstance 模块（DalInstance.pm），LDAP_API 模块（LDAP_API.pm）。DBIs 主要用于日志记录和 Mapping 文件分析，数据库操作等，DalInstance 模块提供了对 Dal_Instance 表的访问和操作接口，LDAP_API 模块主要提供对 LDAP 服务器的访问接口。

ODBC_collector 工作流程如下图 3.9 所示：

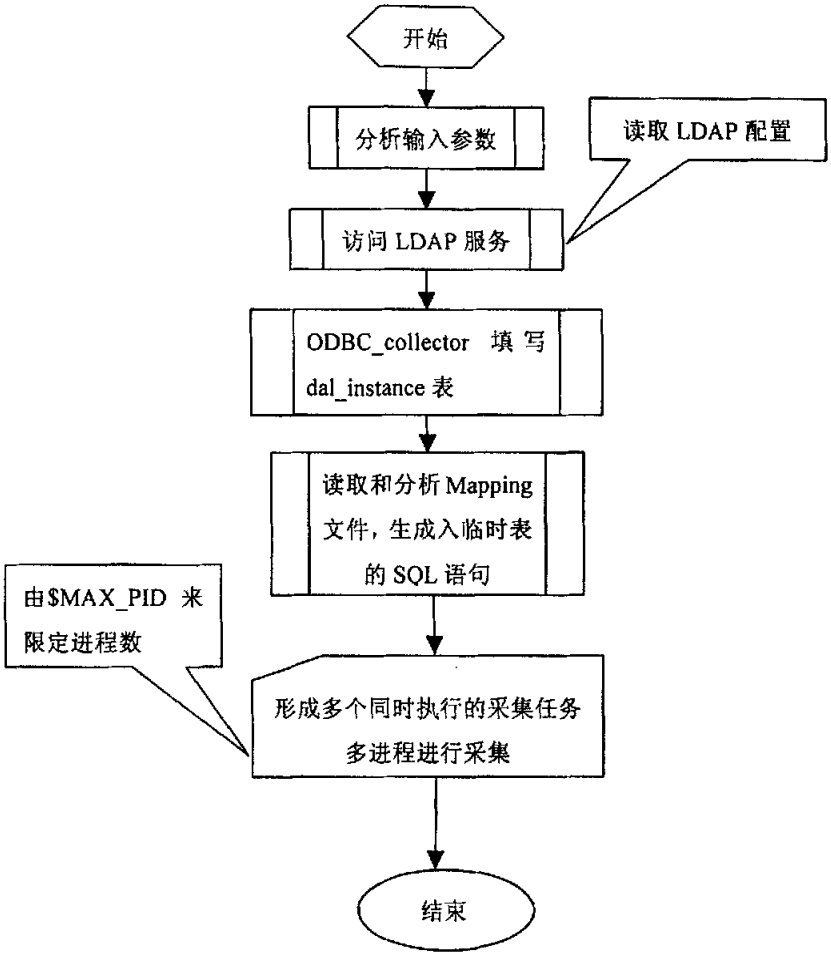


图 3.9 ODBC_collector 工作流程

3. 5. 2. 2 UNIX 下的 ODBC 驱动配置

ODBC（Open Database Connectivity）即开放数据库互连的简称，是 Microsoft 公司提出的一个用于访问数据库的统一界面标准，是应用程序和数据库系统之间的中间件。Windows 系统自带

ODBC 驱动管理器，采集系统采用 UNIX 平台，需要第三方驱动管理器，我们采用 DATADIRECT 公司的产品“DataDirect Connect for ODBC”。在 ODBC 配置上需要注意以下两个文件的应用。

（1） 信息系统文件.odbc.ini

在 UNIX 下要配置数据源，必须在用户主目录下面编辑一个缺省名为“.odbc.ini”的信息系统文件,它也可以在 ODBCINI 这个环境变量中指明全路径和新文件名。文件内容与形式示例如下：

```
[ODBC Data Sources]                    #数据源列表
303=Informix 7.x ODBC Driver           # informix 驱动
404=MERANT 3.60 Oracle Driver          # oracle 驱动
404_txt= MERANT 3.60 Text Driver       # 文件数据库驱动
……

[303]                                  # OMC303 数据源
Driver=/opt/ODBC/lib/ivinf15.so
Description=Informix 7.x ODBC Driver   # informix 数据源
ServerName=omc_sys                     # 数据库服务器的名称
Database=omcdb                         # 主数据库的名称
LogonID=informix
Password=informix

[404_txt]                              #MD 上的文件数据库
Driver=/opt/ODBC/lib/ivtxt15.so
Description=Text driver
Database=$NPM_HOME/data/dest/nokia/404
```

（2） 文件数据库定义文件 QETXT.INI

文件数据库在文件模式采集中有着非常重要的意义。

文件数据库包含一个或者一组文件，一般是 ASCII 文本文件，

一行包含一条记录，以行结束符作为记录分隔符。一条记录区分字段的方式有两种：以分隔符隔开字段，或者为每个字段分配固定的长度。由于待采集的数据字段长短差异很大，且含有各种标点符号字符，所以我们通常采用 TAB 键作为字段分隔符。

文件中包含的数据格式是已知和标准的，程序可以扫描以查询所需要的信息，并作出处理，这比用关系型数据库开销小，速度快，编程灵活性强，目前文本驱动程序不能支持超过 2GB 的文件，我们要处理的文件远远小于这个值。对文件数据库，我们可以执行标准的 SQL 查询、插入、更新、删除操作。

UNIX 平台使用 QETXT.INI 定义文件数据库的结构，通过使用 QETXT. INI 规范，可以定义本地文本数据库中的表。QETXT.INI 定义示例如下：

```
[Defined Tables]
s_tpm_msc.txt = s_tpm_msc
[s_tpm_msc]
FILE= s_tpm_msc.txt
FLN=0
TT= TAB
FIELD1=msc_id,VARCHAR,20,0,20,0,        #NOT NULL
FIELD2=scan_start,VARCHAR,24,0,24,0,    #NOT NULL,DATE
FIELD3=scan_stop,VARCHAR,24,0,24,0,     #NOT NULL,DATE
FIELD4=subscrib_in_vlr,NUMERIC,8,2,8,2,                    #NOT
NULL,DEFAULT 0
FIELD5=fix_ms_call_att,NUMERIC,7,0,7,0,
……
```

说明：

[Defined Tables]：列出所有要定义表名和文件名的关系，本例文件名是 s_tpm_msc.txt，代表文件数据库的表 s_tpm_msc；对于此列表中的每一个表，都需要专门的一节相应说明表的结构。

[s_tpm_msc] 这张表中：

FILE：定义文件名；

FLN：FirstLineNames={0 | 1}，定义文件第一行是列名还是数据，0 代表是数据，1 代表列名即数据从文件第二行开始；

TT：TableType 表格类型，Tab 代表用 TAB 键间隔的表；

FIELD1：代表第一个字段，以此类推。等号后面定义字段名、字段类型、精确度、数值范围等

“#”：为注释开始，关键字 NOT NULL 说明该字段为非空，DATE 说明该字段为标准是间格式‘yyyy-mm-dd hh:mi:ss’ DEFAULT 说明若字段若为空则数据为 DEFAULT 后面的值，其余按照缺省处理。

3. 5. 2. 3 ODBC_collector 调用接口

ODBC_collector 模块的作用是将源数据列表中定义的数据保存到目标数据源。

对于数据库模式采集，源数据是厂家设备数据库，目标数据源是采集服务器数据库 MDDB。

对于文件模式采集，数据流向是“厂家设备→MD 文件数据库→MDDB”，“厂家设备→MD 文件数据库”是个性化编程的过程，获取并处理厂家数据，并存放于 MD 文件数据库，“MD 文件数据库→MDDB”过程中存在 ODBC_collector 调用，源数据列表是 MD 的本地文件数据库，目标数据源是 MDDB 临时表。

Invoker 根据 XML 配置文件调用 ODBC_collector 模块，命令行调用基本形式为：

GP-NPM_odbccollector.pl -o <OSS Label> -task_id <Task_Id> -s <start_time> -e <Stop_time>

-i <instance_id> DalInstance 模块从 Dal_Instance 表中根据任务号和模块名称获得

-ne_type 采集的网元类型

[-ODBC ODBCINI 具体指向]

3．5．3 Adaptor

3．5．3．1 Adaptor 设计

Adaptor 是数据采集层中对采集到的厂家数据进行归一化处理的模块，它的主要功能是按照 Mapping 映射文件定义的数据映射关系，将 ODBC_collector 产生的临时表数据入到 MDDB 的底层性能表中，它与 NPMDB 中的底层性能表一一对应，我们都称之为“底层性能表”，是经过采集系统处理的数据最终呈现形式。

在设计上，与 ODBC_collector 相似，Adaptor 由主程序 GP-NPM_adaptor.pl 和与 ODBC_collector 共用的 Mapping 算法映射文件组成。

工作流程是：

（1） 通过 LDAP 协议获取全局变量和配置信息；

（2） Adaptor 填写 dal_Instance 表，记录采集过程，日志文件；

（3） 主程序 GP-NPM_adaptor.pl 通过读取 Mapping 算法映射文件，分析文件格式与内容，组合 SQL 语句。Adaptor 的实质就是对 MDDB 执行一系列的 SQL 语句，将经过 ODBC_collector 处理过的临时表数据，按照一定标准进入“底层性能表”。

（4） 采集任务并行入库。采集任务并行情况有两种类型：一种是对同一设备采集的数据入 MDDB 中不同的底层性能表；另一种是不同设备数据入同一个底层性能表，需要进行排队入库。

（5） 当临时表中的数据入库到 MDDB,且 Invoker 调用的返回状态为正常时，由 Task_Generator 发性能同步消息给 Task_Manager,由 Task_Manager 生成并启动性能同步任务,再由 Sync_PM 将数据从 MDDB 同步到 NPMDB。

与 ODBC_collector 相同，也使用了三大主要模块 DBIs 模块和 DalInstance 模块，LDAP_API 模块。

工作流程如下图 3.10 所示：

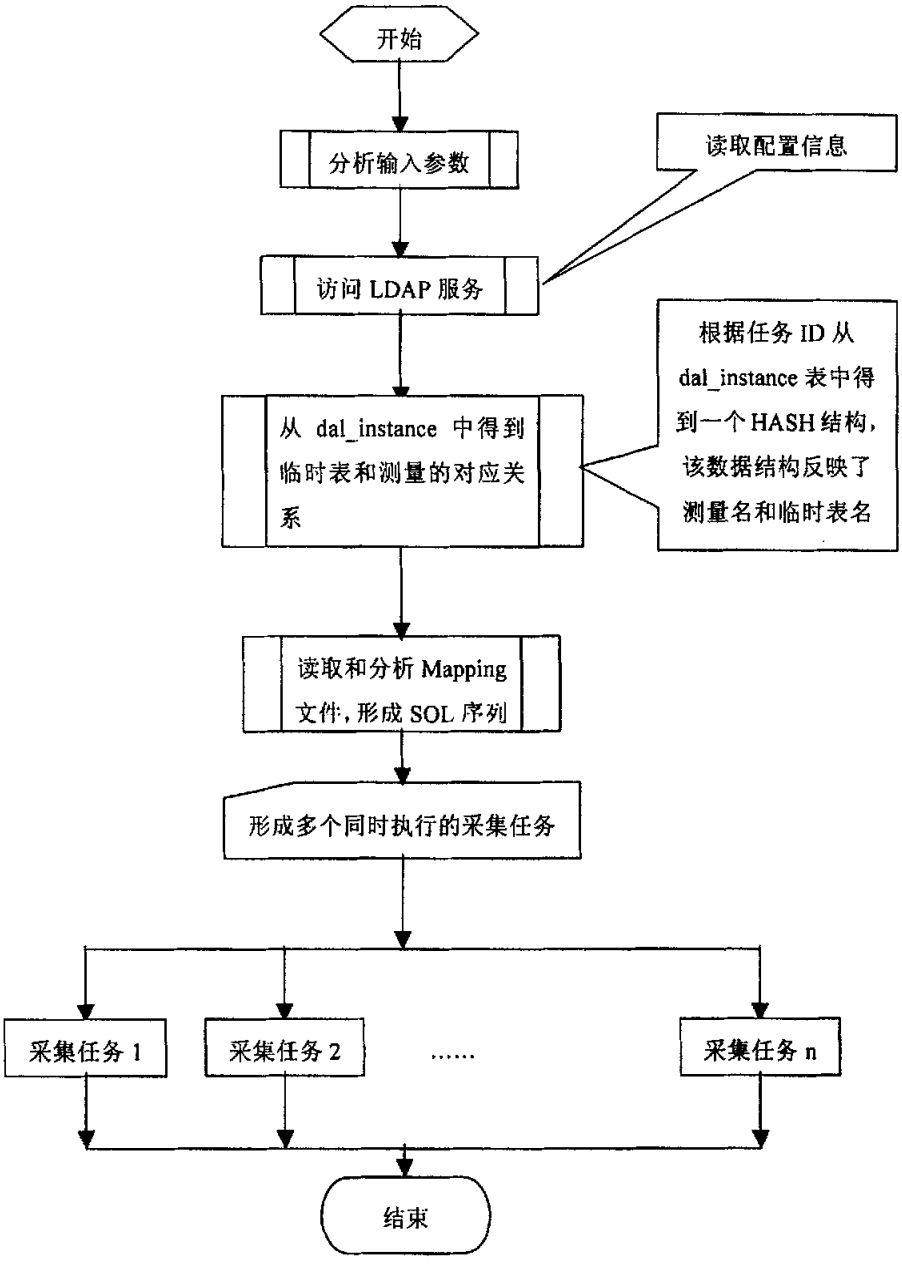


图 3.10　Adaptor 工作流程

3．5．3．2　Adaptor 调用接口

Invoker 根据配置文件调用 Adaptor 模块，命令行调用基本形式为：

GP-NPM_adaptor.pl -o <OSS Label> -task_id <Task_Id> -s <start_time> -e <Stop_time>

-i <instance_id> DalInstance 模块从 dal_instance 表中根据任务号和模块名称获得

-ne_type 采集的网元类型

3．6 MD 数据库表设计

3．6．1 NPMDB 和 MDDB 中底层性能表结构的异同

采集过程中，数据流走向是“厂家设备→MDDB→NPMDB”。“厂家设备→MDDB”完成异构数据归一化处理，执行完成性能采集所有的复杂任务，“MDDB→NPMDB”是一个数据的迁移过程，将底层性能数据从 MDDB 导出到文件，再由文件导入到 NPMDB。两个数据库中的底层性能表，有以下两个异同点：

（1）由于 Sync_PM 性能同步采用文件导出导入不同数据库的方式，因此NPMDB和MDDB中对应性能表结构字段类型和顺序完全一致。

（2）MDDB 和 NPMDB 中性能表的主健和索引因为面向不同应用，所以会不一样。

3．6．2 底层性能表结构

MDDB 和 NPMDB 底层性能表表名相同，字段一一对应，按照 GSM 网络系统结构目前分为以下几类：无线表、交换表、GRPS 相关表、SMSC 短信表、IN 智能网表。最重要的考察项目是无线表和交换表，按照我们数据库的约定，底层表名以 tpd 开头。

（1） 无线表

tpd_radio_bts	记录无线设备 BTS 的性能数据
tpd_cell_ho	记录小区切换性能数据
tpd_radio_bsc	记录无线设备 BSC 的性能数据
tpd_adjacent	记录小区间切入切出信息

（2） 交换表

tpd_msc	记录交换设备 MSC 性能数据
tpd_msc_ho	记录 MSC 的切换性能数据
tpd_destcode	记录目的码信息
tpd_tkgp	记录中继信息
tpd_clear_code	记录清除码信息
tpd_hlr	记录网元 HRL 统计数据
tpd_link	记录 link 链路状态信息

3．6．3 采集控制相关的表

dal_omc_stat	存放 omc 的状态信息
dal_check_err	存放 DataCheck 发现无法采集的任务信息
dal_datacheck	存放 DataCheck 的运行信息
dal_task	存放产生的采集任务。
dal_check_about	存放 OMC 性能表采集的参考记录数

3．6．4 采集过程相关的表

dal_instance	存放 Invoker 调用的层次关系
dal_log	存放采集过程的错误信息
dal_adaptor_result	存放每次运行采集的记录数

3．7　系统环境与开发工具

3．7．1　操作系统

数据采集选择UNIX系统，目前主要是基于SOLARIS 5.6/8 平台上实现。

根据我国 GSM 网目前实际的网络情况，采集系统将网络规模划分四种，并以 SUN 880 (4CPU/4G 内存)作为采集服务器比较基准，量化采集服务器如下：

网络规模	采集记录数	服务器数量（SUN 880）
小型网络	10000 以下	1
中型网络	10000-20000	2
大型网络	20000-40000	3-4
超大网络	40000 以上	4 以上

3．7．2　采集服务器数据库

三期网管中上层应用所使用的数据库称为 NPMDB，它多采用 INFORMIX 或 ORACLE 等大型关系数据库，支持复杂的查询和事务功能。采集服务器数据库从网管中上层应用剥离出来，称为 MDDB，使用它不仅大大减轻了 NPMDB 的负担，并且我们可以根据采集系统自身特点选取更适宜的数据库。

我们选择了 MYSQL 数据库作为 MDDB，主要考虑到数据采集有以下几个方面的特点：

（1） 待采数据量大

一方面，采集数据量大，采集时间频繁，对于一个中型网络，每小时采集的记录数在 10000-20000 之间；另一方面，这些数据没有太复杂的完整性需求，复杂查询访问的需要。MYSQL 作为中小型关系性数据库，减少了很多我们不需要的资源开销如视图、子查询、定义用户类型、触发器、规则和一些事务支持，但提供了速度和维护的容易程度。

（2） MDDB 中数据不做长期存储

首先，MDDB 底层性能表中数据会及时同步到 NPMDB，不用做永久保留；此外，采集过程中出现的大量临时表更无需保存。MDDB 在采集过程中仅对数据做临时性存储，系统脚本会定时清理临时表和临时数据，因此不需要采用大型关系型数据库。

（3） 硬件角度考虑

一个每小时采集记录数在 10000 以下小型网络，以 SUN 880（4CPU/4G 内存）作为一个基准的，需要一台服务器。采集系统独立使用的服务器往往会出现这种情况，当 CPU 快用尽时而内存却大量闲制。MYSQL 数据库有一种表称为 HEAP 表，HEAP 表全部在内存中存取，效率高，但 MYSQL 被关掉或崩溃，就会出现数据丢失，所以非常适合存储临时数据。采集服务器一方面可以提供大量内存，另一方面又在数据库中存取大量临时数据，所以可以充分利用 MYSQL 的内存表特点。创建一个 HEAP 表的 MYSQL-SQL 语句如下：

```
CREATE TABLE email_addresses TYPE=HEAP (
email char(55) NOT NULL,
name char(30) NOT NULL,
PRIMARY KEY(email)
);
```

（4） 平台兼容性

MYSQL 可以在许多平台上工作，既作服务器同时又是客户端，编程方面有我们采集系统需要的 Perl 接口。

3．7．3 编程语言

编程主要使用 Perl 语言，采用 Perl 5.8.3 或以上版本。

Perl，全称是实用摘录与报告语言（Practical Extraction and Report Language），是解释型语言，融合了许多语言的特性。它主要是由 C，其次由 SED、AWK、UNIX SHELL 和至少十数种其他的工具和语言所演化而来。

我们的数据采集系统甚至网管系统的核心中层应用，均采用Perl语言编程，主要因为以下几点：

（1） 强大的文字处理能力

采集系统需要处理大量不同格式与信息的文件，用Perl写处理文字、文档的程序非常方便、简洁。

Perl文字处理功能的核心所在是“正则表达式”。正则表达式赋予了Perl极大的处理和操作自由文本中的“模式”的能力。而在C、C++或Java中实现“正则表达式”的功能非常麻烦，且不容易理解。

（2） 一种强有力的粘合语言

Perl是一门通用的灵活的语言，可以象胶水一样把其他许多不同的模型粘合起来，将SED、AWK 、UNIX SHELL的易用性和编程语言（比如C）的强大功能与可塑性结合起来。它容易调用操作系统功能、服务，方便的进行UNIX系统管理，同时又可以编写复杂的应用程序。作为解释型语言，它写的程序在解释器所能安装的任何系统上都可以运行，而不必担心移植性问题。目前Perl几乎可以在所有的操作系统中运行。

（3） 强大的数据库接口——DBI模块

Perl对数据库的接口称为DBI（Perl Database Interface）。DBI模块是Perl编程语言的标准数据库接口，它独立于数据库，可以与任何相关数据库，如ORACLE、SYBASE、INFORMIX、MYSQL、ACCESS等协同工作。Perl对多种数据库操作都方法一样，操作方便，简单，并且能够直接得到对数据库的操作结果。

（4）丰富的标准模块

模块（module）是Perl中重复使用的代码基本单位。除了DBI模块外，我们应用比较多的其它Perl模块还有：

Net::LDAP：是Perl访问LDAP服务器的应用程序接口，它可以搜索LDAP目录，并可以增加、删除、修改目录及其属性。

IO::Socket：提供Perl对套接字进行处理的应用程序接口。本系统中运用在进程间通讯。

XML::Parser：XML 语法分析器 EXPAT 的 Perl 接口，由 James Clark 用 C 写的一个基于事件驱动（SAX）的非验证性解析器。

XML::Simple ：完全用 Perl 语言写成，它是作为 XML:: Parser 模块之上的一个 API 层实现的，用于读写 XML 的普通 API。XML::Simple 能够解析一个 XML 文件并以一个 Perl 哈希引用返回数据。

Net::Telnet：用于 TELNET 协议通信。

Net::FTP：用于 FTP 协议通信。

3．7．4 LDAP 服务器

我们的采集系统使用的 SUN 公司的 iPlanet Directory Server。iPlanet Directory Server 是一个功能强大、伸缩自如的分布式目录服务器，以符合业界标准的轻型目录访问协议 （LDAP 协议）为基础。它支持所有得主流平台如 Solaris、HP-UX、AIX、Windows 等

iPlanet 目录软件开发工具包（SDK），它允许开发人员使用 C、Java、JavaScript、Perl 以及其它编程语言，在所有流行平台上创建 LDAP 应用。

3．7．5 ODBC 驱动管理器

NPM 采集服务器使用的第三方 ODBC 驱动程序是 DataDirect 公司的产品“DataDirect Connect for ODBC”。它所采用的“Wire protocol” 设计避免了对数据库客户端软件的需求，简化了安装和管理过程，而且动态的提高了程序性能。

DataDirect Connect for ODBC 支持所有主要数据库，如 Oracle，DB2，SQL Server，Sybase，Informix； 支持所有主要平台 Windows，Unix，Linux。

3．8 本章小节

以某省项目需求规范为例，用户针对数据采集提出“及时性、

完整性和一致性”的要求如下：

- 系统能够及时完整地采集每小时（连续 24 小时）的性能数据；
- 在网元或 OMC 的原始性能数据产生后，从采集到呈现不大于 45 分钟；
- 系统应保证数据采集的完整性，采集完成后网管系统与采集点（网元/OMC）的数据要保持一致。

结合用户需求与本章第一节所提出的采集系统要解决的问题，网管三期数据采集系统成功的完成了如下转变：

（1） 实现两个分离，提高系统性能

采集系统通过将网管底层采集数据库与中上层应用数据库分离，NPM 性能采集和 NRM 资源采集系统分离，增加采集控制部分等设计，为采集系统提升运行效率提供了有力保障，并且大大减少了对网管系统不同应用之间的干扰和影响。

（2） 变被动采集为主动采集

采集方式以前是维护人员定时设定采集时间，采集行为被动，采集及时性不能保证。以前为了保证考核数据的完整性，维护人员设定数据采集时间较晚，用户一直以来都是当日考察昨日数据，以牺牲数据及时性来确保完整性。三期采集系统，转变为采用 DataCheck 模块以轮询方式进行数据检测，在数据完备后，立刻进行采集，并在采集完成后，主动触发数据汇总等网管中上层应用，为用户提供各类报表。DataCheck 采用的是主动采集方式，从采集到呈现目前已经能实现 15 分钟的保证。

（3） 系统更加智能化，减少维护量

三期系统，采集时间无需再依赖维护人员的经验设定，而且配置文件由 LDAP 服务器进行统一管理，停用了人工备份配置文件的方式。这样，不仅减少了人工维护内容，更重要的是减少了人为误判断或误操作对采集系统的造成的不利影响。

4 实例分析

在采集核心系统搭建完成后，需要根据运营商的实际网络设备情况，开发不同设备的数据采集。网管三期工程中标移动、联通 GSM 网 54 省项目，不同地区运营网络会采用不同厂商以及版本的通信设备，但是这些通信设备从总体来说主要集中于几个国外和国内厂商，如爱立信、西门子、诺基亚、摩托罗拉、贝尔、阿尔卡特以及中兴、华为、康维、东信北邮等，设备版本在全局来看也比较集中于一个厂商的几个版本。基于以上特点，项目组成员按照设备厂商进行分工，一个开发人员负责一个或几个厂商的性能管理（NPM）和资源管理（NRM）数据的所有采集开发。作者本人在职期间负责全网项目中诺基亚和摩托罗拉两厂商所有通信设备的性能和资源数据采集开发，下面就以诺基亚 GSM-NMS2000 T12 网络管理系统采集为例进行设备采集实现分析。

4．1 采集背景

4．1．1 设备环境

NOKIA NMS2000 T12 版本 OMC 主机的操作系统为 HP-UX，NMS2000 数据库采用 ORACLE。诺基亚 GSM-NMS2000 管理 GSM 网络所有的 NOKIA 设备，包括交换、无线、短信、GPRS 等。本文以采集交换数据的底层性能表 tpd_msc 为例进行介绍，其数据采集通过 ODBC 连接 NMS2000 数据库和人机指令两种方式进行，涵盖了数据库模式和文件模式两种采集方式。

4．1．2 厂家数据库及人机指令与网管底层性能表对应关系

（1） 厂家数据库与底层性能表对应关系

如下表 4.1 所示，一张底层性能表 tpd_msc 的数据来源于厂家数据库六张表。

厂家数据库表名	网元粒度	NPM 数据库表	备注
p_msc_trafficability	MSC	tpd_msc	自身为 MSC 粒度
p_msc_vlr	MSC		自身为 MSC 粒度
p_msc_tc	MSC		自身为 MSC 粒度
p_msc_nd	目的码，并区分目的码的方向和类型		需汇总至 MSC 粒度
p_msc_load	UNIT		网元粒度 UNIT，需汇总至 MSC 粒度
p_msc_cc_calls	CLEAR		网元粒度为结束码

表 4.1 厂家数据库与底层表对应关系示例

（2） 人机指令对应关系

通过在设备主机发指令得到文件，进行格式分析，提取所需数据，采入 tpd_msc 表。

对应人机指令或文件名	NPM 数据库表	指令
ZMVF:::DETACHED=N:;	tpd_msc	可获取 active_subs
ZTUF:MTC;		可获取 total_called_ans
ZMVI; ZMVF:::HLR=<local hlrs>;		可获取 vlr_roam_sub
ZMVF:::DETACHED=Y; ZMVF:::DETACHED=N;		可获取 subscrib_in_vlr
ZTUL:FAC;		可获取 total_called_ans

表 4.2 人机指令与底层表对应关系

人机指令格式说明：

exemmlmx -c "ZTUF:MTC;" -i 62 –f filename

注：

exemmlmx 是 NOKIA OMC 主机提供的取性能数据指令，结果以文件方式呈现。

-c 为指令名称

-f 为批命令的文件名

-i 为网元在 NMS2000 数据库中 objects 表的 int_id，即所采集网元的在厂家的 ID。

远程发送指令的方式：rsh -l $user $host exemmlmx -f $remotecmd -i $neid >> $logfile

$user：OMC 上为网管创建的用户名；

$host：OMC 主机地址；

$remotecmd：批处理文件，一行一个 exemmlmx 命令；

$neid：网元在厂家端的 ID 编号；

$logfile：发布 exemmlmx 指令后生成的结果文件

4．2 采集设计举例

4．2．1 tpd_msc 映射算法

一张底层性能表由两部分信息组成，一类是基本信息，反映设备基本配置信息，以及所采数据时间，这一部分所有底层性能表都具有；另外一类是若干具体指标信息，即实际采集的考核数据，根据运营商的规范统一做出，取不同设备采集的并集做出包含内容最广的表结构。

4．2．1．1 性能表基本信息

下表 4.3 所示不同底层性能表所共有的基本信息。

PK: record_id

Unique:(ne_id,scan_start_time,scan_stop_time)

列名	描述	数据类型	约束	算法
object_rdn	一个 MSC 的标志名	varchar (255)		$OMC_ID||':'||'101'||':'||p_msc_tc.msc_id
ne_id	MSC 在 MDDB 的 objects 表中唯一标识 int_id	integer		使用 CRC32 算法通过 object_rdn 计算可得
record_id	一条性能记录的唯一标识	integer		系统自动生成
scan_start_time	所采集数据的起始时间	datetime year to second	not null	p_msc_tc. period_start_time
scan_stop_time	所采集数据的结束时间	datetime year to second	not null	p_msc_tc.period_stop_time
omc_id		integer		$OMC_ID
vendor_id		integer		4(代表 NOKIA 厂商)

表 4.3 tpd_msc 表基本信息

每一个网元具有唯一的字符串标识 object_rdn 和整型标识 ne_id，并且这两个标识采用 CRC32 算法一一映射。

4. 2. 1. 2 数据库模式采集字段选例

表 4.4 列举了需要通过数据库模式采集的一部分字段。对于表 tpd_msc 的每个字段，算法列指出可以使用厂家数据库中哪些表中的哪些字段，以及做什么样的运算可以获得。

列名	描述	数据类型	算法
att_calls_sys	系统试呼次数	integer	p_msc_trafficability. tot_calls
suc_answ_sys	系统应答次数	integer	p_msc_trafficability. answered
effect_calls	有效呼叫次数	integer	p_msc_trafficability.tot_calls-((p_msc_cc_calls.calls_in_signal+p_msc_cc_calls.calls_in_ring) where p_msc_cc_calls.clr_code_id in (10,300,301,304,306,307,30A,30B,311,312,313,318,817,C02,C01))
net_total_succ	网络接通次数	integer	p_msc_trafficability.answered+((p_msc_cc_calls.calls_in_signal+p_msc_cc_calls.calls_in_ring) where p_msc_cc_calls.clr_code_id in (5,6,12))
att_pages	系统寻呼次数	integer	p_msc_vlr.succ_page+ p_msc_vlr.fail_page
att_calls_msc	交换机试呼次数	integer	SUM(p_msc_tc.nbr_calls_tc) where traffic_category in (5,6)
msc_proc_max_load	MSC 处理器峰值负荷(指主处理器)	float	MAX(p_msc_load.proc_peak_load)/10 Group by int_id
……	……	……	……

表 4.4　tpd_msc 表部分字段（数据库模式）

4. 2. 1. 3 文件模式采集字段选例

文件模式采集会遇到形式多样的文件报告，程序员使用 Perl 语言强大的文档、文字处理能力，解析获取的文件报告，得到待采数据，并最终形成 MD 文件数据库中的格式化文件

表 4.5 列举了表 tpd_msc 需要通过文件模式采集的四个字段。这些字段的是从不同或相同的文件中得到的。

不同设备出文件的形式无一类似和相同，就是对同一设备发指令取得的文件的方式和呈现形式也根据所取的数据指标不同有很大差别，本文仅以 active_subs 举例，展示其文件报告形式。

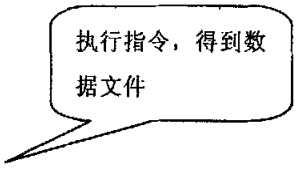
列名	描述	数据类型	算法
active_subs	VLR 开机用户数	integer	参见 FIELD
total_called_ans	被叫呼通次数 是指 GSM 用户做被叫时，交换机对用户发出的振铃总次数。	integer	略
vlr_roam_sub	VLR 漫游用户数	integer	略
subscrib_in_vlr	VLR 用户总数	integer	略

表 4.5 tpd_msc 表部分字段（文件模式采集）

active subs 字段算法关系如下所示：

FIELD active_subs

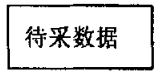
FIELD DESCRIPTION:
VLR 开机用户数
COMMAND:
ZMVF:::DETACHED=N;



REPORT FORMAT（文件形式如下）：

ZZMVF:::DETACHED=N:;
LOADING PROGRAM VERSION 5.43-0

VLRU COUNT
VLRU 0: 9168
VLRU 1: 9359
VLRU 2: 9180
VLRU 3: 9129
VLRU 4: 9126



Total: **45962** (77.0%)

COMMAND EXECUTED

EXAMPLE:
45962

4．2．2 整体设计分析

具体设备采集开发与采集核心系统整合关系如下图 4.1 所示：

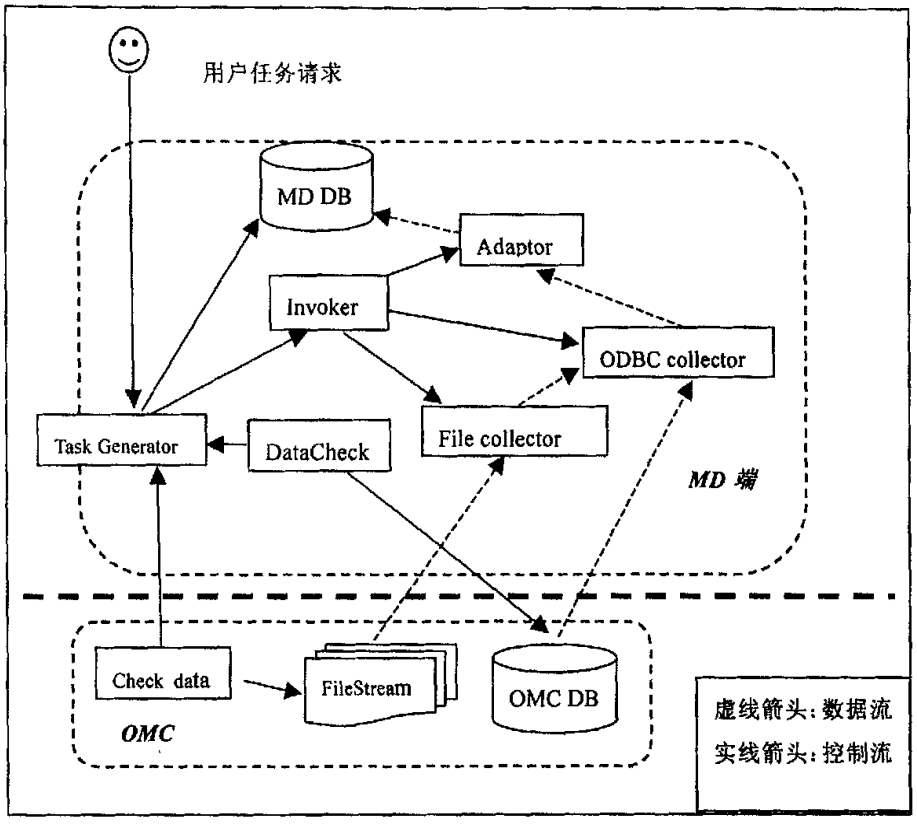


图 4.1 设备采集实现机制

每添加一版本设备采集，我们要做的工作有两方面。第一：通过标准的配置及映射文件整合入采集核心系统；第二，由于采集平台的出现就是为了处理厂家设备的异构问题，进而为中上层应用提供统一平台，所以需要程序员针对不同厂商不同版本设备做独立编程工作，解决的问题多与文件模式采集部分相关。

从上图 4.1 中虚线所表示的数据流上看，数据从 FileStream（文件）经过 File_collector 过程进入文件数据库，在接受 ODBC_collector 模块处理前，需要对设备进行个性化编程工作；数据从 ODBC_collector 到 Adaptor 直到最后入 MDDB 是一个标准化过程，需要按规范制作算法映射文件，即 ODBC_collector 和 Adaptor 共同使用的具有标准格式 Mapping 文件。

4. 2. 3 LDAP 配置设计

4. 2. 3. 1 LDAP 树图

假设实例中 OMC 的字符串标识为 401，要配置 OMC401 的采集，需要在 LDAP 目录添加一定数目节点，如下图 4.2 所示：

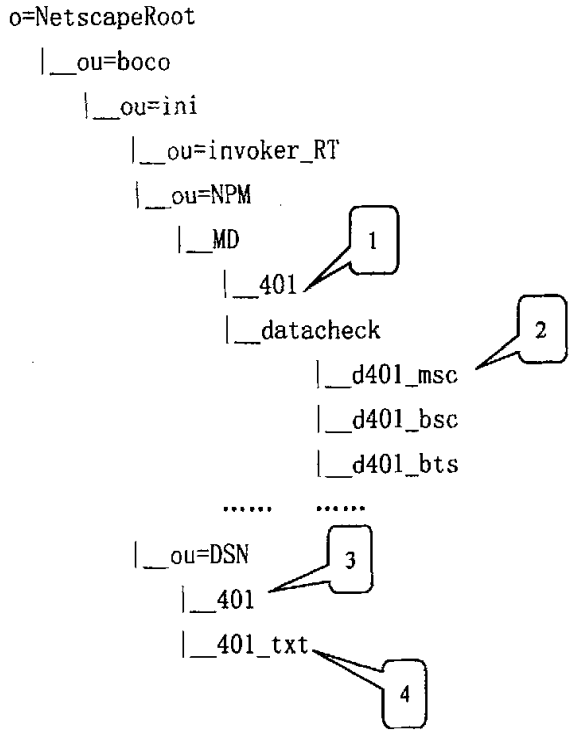


图 4.2 OMC401 的 LDAP 树图

图中标识的节点①到节点□将在下文中介绍。

4. 2. 3. 2 节点①配置

节点①是 OMC 类型节点，在 LDAP 树图中的位置是：

ou=401，ou=MD,ou=NPM,ou=ini,ou=boco,o=NetScapeRoot

参数配置如下：

[401_0]

objectclass = omc

dsn_list = 401

mddb_dsn = mddb

vendor_name = nokia

backup_dsn = backupdb

oss_txt_dsn = 401_txt

col_type = OMC

pm_backup_mapping = 0

pm_local_mapping = 0

pm_asm_list =
TPD_ADJACENT_NOKIA|TPM_MSC_NOKIA|TPD_HLR_NOKIA|TPM_TKGP_NOKIA|TPD_LINK_NOKIA|TPM_DESTCODE_NOKIA|TPD_RADIO_BSC_NOKIA|TPM_RADIO_BTS_NOKIA|TPD_CELL_HO_NOKIA|TPD_CLEAR_CODE|TPD_MSC_HO_NOKIA

pm_common_mapping =
/opt/BOCO.DAL/NPM/mapping/nokia/release/DC3.0_NPM_cmcc_nokia_gsm_t12_odbc_mysql.map

omc_host = 10.19.8.0

omc_uid = omc

omc_pwd = omc

csv = 0

omc_id = 401

BASE_NODE =
ou=MD,ou=NPM,ou=ini,ou=boco,o=NetScapeRoot
cook_obj_class = 100
oss_db_dsn = 401
flag_backup = 0
vendor_id = 4
version = t12
OU_NAME = 401
pm_local_mapping = 0
data_source =
/opt/BOCO.DAL/NPM/data/source/nokia/401
data_dest =
/opt/BOCO.DAL/NPM/data/dest/nokia/401

说明：

pm_asm_list 控制需要对 OMC401 做哪些底层表采集，如“TPD_ADJACENT_NOKIA|TPM_MSC_NOKIA|TPD_HLR_NOKIA”分别代表底层表“tpd_adjacent|tpd_msc|tpd_hlr”。

pm_backup_mapping 、pm_local_mapping 这些等于 0 的值是为将来系统完善和升级预留的扩展参数，目前未赋值。

data_source、data_dest 是为此设备独立编程添加的参数。在对设备采集程序进行独立编制的过程中，我们需要用到一些参数的传入传出，开发过程提倡由 LDAP 数据库做统一的参数管理。

4. 2. 3. 3 节点②配置

节点②是 Datacheck 类型节点，在 LDAP 树图中的位置是：

ou＝d401_msc,ou=datacheck,ou=MD,ou=NPM,ou=ini,
ou=boco,o=NetScapeRoot

参数配置如下：

[d401_msc_0]

objectclass = datacheck

check_id = d401_msc

config_file = nokia_msc.xml

invoker_cmd = GP-NPM_invoker.pl -task_name PMS -setodbc

OU_NAME = d401_msc

omc_id = 401

restart_cmd = DataCheck.pl -i d401_msc

BASE_NODE = ou=datacheck,ou=MD,ou=NPM,ou=ini,ou=boco,o=NetScapeRoot

logfile = /opt/BOCO.DAL/NPM/trace/datacheck/d401_msc.log

---

4. 2. 3. 4 节点③配置

节点③是DSN类型节点，是OMC401数据库模式采集的ODBC数据源信息，在LDAP树图中的位置是：

ou=401,ou=DSN,ou=ini,ou=boco,o=NetScapeRoot

参数配置如下：

[401_1]

objectclass = dsn

db_name = 401_db

db_user = omc

db_pwd = omc

OU_NAME = 401

db_dsn = 401_db

BASE_NODE =

ou=DSN,ou=ini,ou=boco,o=NetScapeRoot

db_type = ODBC

---

4. 2. 3. 5 节点④配置

节点④是 DSN 类型节点，是 OMC401 文件模式采集中使用的 MD 文件数据库数据源信息，在 LDAP 树图中的位置是：

ou=401_txt,ou=DSN,ou=ini,ou=boco,o=NetScapeRoot

参数配置如下：

[401_2]

objectclass = dsn

db_name = 401_txt

db_user = 0

db_pwd = 0

OU_NAME = 401_txt

db_dsn = 401_txt

BASE_NODE =
ou=DSN,ou=ini,ou=boco,o=NetScapeRoot

db_type = ODBC

---

4. 2. 4 Invoker 配置

每增加一类设备采集，就需要在 GP-NPM_invoker.xml 配置文件中增加有关采集过程实现的信息，并统一由 Invoker 主程序解析。NOKIA NMS2000 T12 版本 OMC 的采集配置如下：

<invoker>

<task_def>

<vendor id="4" name="nokia" version="t12" type="OMC"

```
task="PM">
        <step id="1" action_name="odbccollector" onError="Exit"
retry="" sleep="" timeout="3600"/>
        <step id="2" action_name="adaptor" onError="Exit"
retry="" sleep="" timeout="3600"/>
    </vendor>
    <vendor id="5" name="nokia" version="t12" type="OMC"
task="PMS">
        <step id="1" action_name="nokia_s_catfiles"
onError="Exit" retry="" sleep="" timeout="3600"/>
        <step id="2" action_name="odbccollector" onError="Exit"
retry="" sleep="" timeout="3600"/>
        <step id="3" action_name="adaptor" onError="Exit"
retry="" sleep="" timeout="3600"/>
    </vendor>
  </task_def>
  <action_def>
    <action name="nokia_s_catfiles">
    <command>$NPM_HOME/mbin/nokia/Nokia_t12_pm_catfiles.
pl</command>
    <arg_list>
      <argument id="1" name="s" optional="1" from="s">$[From
Opt]</argument>
      <argument id="2" name="o" optional="1"
from="o">$[From Opt]</argument>
      <argument id="3" name="d" optional="0"
from="">$LDAP::data_dest</argument>
      <argument id="4" name="task_id" optional="0"
from="task_id">$TASK_ID</argument>
    </arg_list>
```

```
        </action>
    </action_def>
</invoker>
```

4. 2. 4. 1 <task_def>节

<task_def>为各个厂家不同设备、不同采集类型作出定义。每一类采集任务使用一个<vendor>节点。

（1） <vendor>节点

name，version，type 和 task 四个属性唯一确定一种采集任务。

<vendor>节点各属性意义是：

id：唯一性标识；

name：厂家名称；

version：设备版本号；

type：采集的类型，例如是 OMC 还是直联其它网元方式；

task： 辅助区分同一设备的不同采集方式。

NOKIA NMS2000 T12 设备同时存在两种采集方式，Invoker 调用接口使用-taskname 参数进行区别，体现在配置文件中是<vendor>节点的 task 属性。

采集任务一： name="nokia" version="t12" type="OMC" task="PM"。

调用命令 GP-NPM_invoker.pl [-task_name PM]，（不接-task_name，默认 PM 任务）。

进行数据库模式采集，依次执行两个标准动作（action）ODBC_collector、Adaptor。

采集任务二： name="nokia" version="t12" type="OMC" task="PMS"。

调用命令 GP-NPM_invoker.pl -task_name PMS。

进行文件模式采集，首先执行自定义动作 nokia_s_catfiles，再依次执行标准动作 ODBC_collector、Adaptor。

（2） <step>节点

<step>定义采集步骤，各属性意义是：

id：任务执行顺序标识；

action_name ：采集行为（action）名称，详细的行为定义在<action_def>子节点；

onError：该行为出错后的处理，目前提供两种方法 Exit（终止任务），Ignore（忽略错误，继续执行）；

timeout：该行为执行的最大时长，当该行为执行超过指定的值后做超时处理；

retry：超时后重试的次数；

sleep 定义了每次重试之间的时间间隔，当几次重试都失败后，就认为该行为执行失败。

4. 2. 4. 2 <action_def>节点

<action_def>节点定义具体的采集行为，这些采集行为有的是公共行为，比如 ODBC_collector、Adaptor 是每个采集都会经历的步骤，而有的是为设备独立设计的。

（1） <action>节点

<action>节点确定执行此采集行为的程序路径以及参数传入信息。

name：采集行为名称，对应<step>节点<action_name>；

<command>：命令行；

<arg_list>：命令行参数列表详细定义在其子节点<argument>。

（2） <argument>节点

name：参数名称；

optional：表示该参数是否可选

from：指明该参数继承自 GP-NPM_invoker.pl 命令的哪个参数。比如 from="s"，参考 GP-NPM_invoker.pl -o <OSS Label> -task_id <Task_Id> -s <start_time> -e <stop_time>可知，代表

<start_time>参数。

在<argument>的文本中可以灵活的定义一些变量，规则如下：

$TASK_ID：是一个内置的关键字，表示采集的任务号，他会被自动替换成本次采集的任务号；

$LDAP::data_dest：表示该变量出自 LDAP 目录的 OMC 类型节点，每个采集任务只对应一个 OMC 节点，不会出现混淆；

$[from opt]：参数继承自 GP-NPM_invoker.pl 命令行；

其余的变量均来自环境变量。

如示例中的 action“nokia_s_catfiles”命令调用由 xml 文件拼接可得：$NPM_HOME/mbin/nokia/Nokia_t12_pm_catfiles.pl –s <scan_start_time> -e <scan_stop_time> -d <data_dest> - task_id <task_id>。

4．2．5 DataCheck 厂家模块设计

DataCheck 采用“核心程序+厂家模块”的方式检测设备的数据完整情况，对于本文所谈及的 OMC401 底层交换表 tpd_msc 的数据源检测，调用命令是“DataCheck.pl –i $d401_msc”。d401_msc 是 LDAP 目录的 Datacheck 类型节点 OU 名，在节点参数中定义了 d401_msc 所关联的 XML 文件。每个厂家设备模块都有两个入口。

（1） XML 配置文件

每类设备特定的采集有一个配置文件，DataCheck 主程序解析 XML 文件获得厂家包文件位置，以及配置信息。底层交换表 tpd_msc 对应配置文件 nokia_msc.xml，内容如下：

```
<DataCheck>
    <Lib    path="common/modules/vendors/nokia/datacheck" />
    <Pkg    pname="nokia_t11_pm_datacheck.pl" />
    <time          heartbeat="60"    check="60"    next="+1h"
timeout="5400"   limit="-24h" />
    <pm_table   limit ="0.8" name="tpd_msc" />
```

```
    <table num="4" >
        <source id="1" tname="p_msc_vlr" />
        <source id="2" tname="p_msc_tc" />
        <source id="3" tname="p_msc_load" />
        <source id="4" tname="p_msc_trafficability" />
    </table>
</DataCheck>
```

说明：

<Lib>：定义厂家包程序路径；

<Pkg>：指出厂家包程序名称；

<time>： 时间参数。属性 heartbeat 指厂家模块的通讯心跳间隔, check 指检查数据源的时间间隔,next 是指采集任务的标准时间间隔,timeout 是指检测任务的超时时间，limit 是指历史任务的超时时间；

<pm_table>：属性 name 定义该采集任务对应的底层性能表列表，limit 定义数据完整性的判断比例（大于前三次采集数目平均的 xx%,首次采集不判断这个）。

<table>：检查的数据源表 SQL 定义，对厂商数据库执行的 SQL 语句，访问结果的记录数作为判断依据。

（2）厂家程序包

厂家包里面必须包括两个函数：data_check 和 check_omc_stat，供 DataCheck 主程序 DataCheck.pl 调用。

a） data_check

data_check 的输入输出是：

input： (1) vendor config hash

input： (2) ldap config hash

input： (3) start_time

input： (4) check_id（datacheck 类型节点）

input： (5) ne_list

return： check hash :status,start_time,stop_time

status=1 数据准备好；

status=0 数据未准备好。

input 的参数由主程序通过读取 LDAP 数据库配置得到。

b）check_omc_stat

check_omc_stat，检测 OMC 的状态，输入输出是：

input：(1) check_id

input：(2) ldap_config

return:：result

0 OMC 状态正常；

-1 OMC 不能链接。

4．2．6 File_collector 与 Parser 模块

数据采集系统在整个网管系统中的作用，就是处理通信设备的异构问题，将不同设备的异构数据进行归一化处理。File_collector 与 Parser 作为文件模式采集拥有的程序模块，完成数据从“厂家设备→MD 文件数据库”的处理。由于不同设备数据产生机理不同，File_collector 在对设备主机进行文件获取上，也有着不同的处理方式，视具体设备而定，所以通常由程序员根据厂商提供的设备算法文档自行开发。当取得设备文件后，Parser 即文件解析模块负责分析文件，并从文件中获取我们需要的数据指标。

File_collector 与 Parser 是两个典型的面向设备异构性的程序模块，由于独立性强，一般程序员独立编制程序完成。在 NOKIA NMS2000 T12 版本设备采集实例中，我既在采集服务器放置程序进行文件获取与解析，又在对方设备主机上放置了必要的程序文件。考虑到在对方主机上进行作业，设计原则是尽量保持少放或不放置程序在对方主机，主要的调用以及处理工作由采集服务器完成。

4．2．7 ODBC_collector、Adaptor 厂家模块

4．2．7．1 ODBC 配置

下图 4.3 显示了从 OMC 到 MD 的数据流走向。

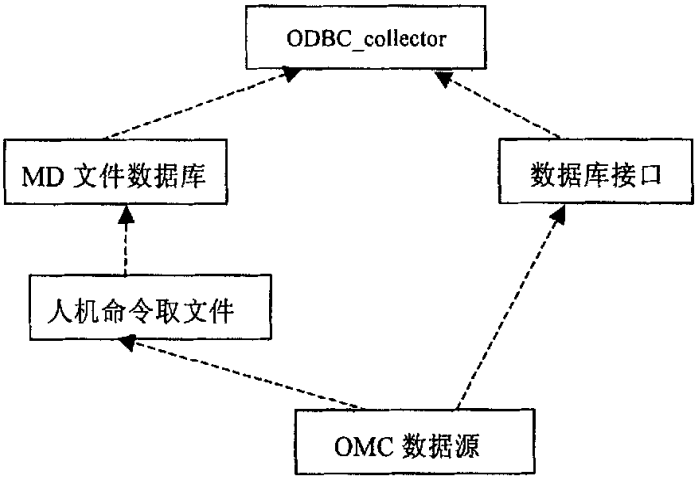


图 4.3 数据流走向图（OMC 到 MD）

数据库接口需要进行 ODBC 配置，在信息系统文件.odbc.ini 添加 OMC401 的数据源信息。

NMS2000 T12 OMC 关于 tpd_msc 表的采集，有四个字段是通过文件模式采集得到。由发指令方式取得的设备文件经过 File_collector 与 Parser 模块处理后，我们需要的最终数据结果存入为 OMC401 创建的文本数据库中，其数据库结构定义文件 QETXT.INI 内容示例如下：

[Defined Tables]

s_msc_pm.txt = s_msc_pm

s_hlr_pm.txt = s_hlr_pm

[s_msc_pm]

FILE=s_msc_pm.txt

TT=TAB

FLN=0

FIELD1=msc_rdn,VARCHAR,10,0,10,0,

FIELD2=start_time,VARCHAR,40,0,40,0,

FIELD3=active_subs,NUMERIC,10,0,10,0,

FIELD4=total_called_ans,NUMERIC,10,0,10,0,

FIELD5=vlr_roam_sub,NUMERIC,10,0,10,0,

[s_hlr_pm]

FILE=s_hlr_pm.txt

TT=TAB

FLN=0

FIELD1=hlr_rdn,VARCHAR,10,0,10,0,

FIELD2=start_time,VARCHAR,40,0,40,0,

FIELD3=subscrib_in_hlr,NUMERIC,10,0,10,0,Mapping 文件

表 s_msc_pm 内容如下表 4.6：

列名	描述	数据类型
msc_rdn	网元 Msc 的标志	VARCHAR
start_time	开始时间	VARCHAR
active_subs	VLR 开机用户数	NUMERIC
total_called_ans	被叫呼通次数	VARCHAR
vlr_roam_sub	VLR 漫游用户数	VARCHAR

表 4.6 s_msc_pm 文件表结构

4. 2. 7. 2 厂家 Mapping 文件编写

ODBC_collector 模块操作的数据源有厂家设备数据库、MD 文件数据库、MDDB，Adaptor 模块操作的数据源是 MDDB。ODBC_collector、Adaptor 共用一个算法映射文件(Mapping 文件)，其主程序解析算法文件，获得对数据库的操作信息。采集任务所使用的 Mapping 文件，定义在 LDAP 树图 OMC 类型节点的 pm_common_mapping 参数中。

Mapping 文件内容按小节编排,对操作流程有主要影响的小节是：[REPORT]、[MEASUREMENT]、[MSNTCOL]、[性能采集名_STEP]、[性能采集名_STEP_BEFORE]、[性能采集名] 。

1） [REPORT]

Report 节是性能采集的入口，标题和内容字段之间由 tab 间隔，程序按行读取数据。

```
[REPORT]
#=============================================================
#!report                    tpdtable
#=============================================================
TPM_MSC_NOKIA                  tpd_msc
TPM_RADIO_BTS_NOKIA      tpd_radio_bts
…… …… ……
#=============================================================
```

列名	含义
report	性能采集名，命名规则：[性能目标表名]_厂家代号（全大写），例如：TPM_MSC_NOKIA，在 MDDB 中表现为一个临时表，收集 NOKIA 的 tpd_msc 的采集有关数据；
tpdtable	底层性能目标表，例如：tpd_msc

2) [MEASUREMENT]

MEASUREMENT 用于定义从厂家数据库中采集哪些测量（即厂家数据库表）到 MDDB，每一个测量对应 MDDB 中一个临时表，该临时表中保存了从厂家数据库中的对应表获取的性能数据。

```
[MEASUREMENT]
#=============================================================
#!report      ossdsn      msnt          msnttab      shour    condition
#=============================================================
TPM_MSC_NOKIA        OSS_DB_DSN    p_msc_all    p_msc_tc
    to_char(period_start_time,'HH24')                              1=1   group
by int_id||'',trunc(period_start_time,'HH24'),trunc(period_stop_time,'HH24')
TPM_MSC_NOKIA        OSS_DB_DSN    p_msc_vlr    p_msc_vlr
    to_char(period_start_time,'HH24')
…… ……
TPM_MSC_NOKIA        OSS_TXT_DSN   s_msc_pm     s_msc_pm
…… ……
#=============================================================
```

列名	含义
Report	性能采集名，对应一个临时表，在 Report 段中进行定义
ossdsn	数据源名。对应 LDAP 目录中 OMC 类型节点中的参数。
Msnt	测量名，对应 MDDB 中一个临时表。此处定义的每一个测量都要在后面的[MSNTCOL]小节指定测量需要采集的字段。
msnttab	厂家数据库中的表名，我们需要从这些表中获取性能数据
Shour	提取厂家时间字段中的“小时”。
condition	采集条件。从厂家数据库获取数据的 SQL 语句的条件。

3) [MSNTCOL]

MSNTCOL 小节用于细化 MEASUREMENT 小节中定义的测量，该小节主要定义了每一个测量所对应的表需要从厂家数据库表中获取哪些字段，以及该测量临时表中的字段与厂家字段的映射关系。性能采集核心程序完成对 MEASUREMENT 和 MSNTCOL 的分析和处理之后，每一个测量对应一个临时表。

```
[MSNTCOL]
#=============================================================
#!msnt                  column          arith
#=============================================================
#----------------------------TPM_MSC_NOKIA--------------------------------
p_msc_all|index|starttime   scan_start_time   trunc(period_start_time,'HH24')
p_msc_all|index|stoptime    scan_stop_time    trunc(period_stop_time,'HH24')
...... ......
s_msc_pm                    active_subs       active_subs
s_msc_pm                    total_called_ans  total_called_ans
...... ......
#=============================================================
```

列名	含义
msnt	采集测量名，测量名在 MEASUREMENT 段中的 msnt 列进行定义。其它几个标志意义： starttime　测量开始时间标志 stoptime　测量结束时间标志 index　在测量临时表中建立索引的字段
column	测量临时表的字段名，此处定义的字段名将在性能数据入库的过程中得到使用。
arith	测量算法，对应厂家字段或者经过计算后的厂家字段

4） [性能采集名_STEP]

性能采集名是指[REPORT]小节的 report 列定义的采集名，用于定义数据从测量临时表到性能目标表的入库过程。

```
[TPM_MSC_NOKIA_STEP]
#=================================================================
#!Step   StepBefore   ChkSpilth   Stepactive   Measurement
    WHEREGROUP
#=================================================================
Step001      0        0        INSERT        p_msc_all    group    by
scan_start_time,scan_stop_time,msc_rdn,old_rdn
Step01       0        0        UPDATE        p_msc_vlr
Step011      1        0        UPDATE        s_msc_pm
…… ……
#=================================================================
```

列名	含义
Step	采集步骤名，自己定义。
StepBefore	设为 1，执行该步骤之前需要执行指定的 SQL 语句对测量临时表进行处理，如果设为 0，则不需要
ChkSpilth	设为 1，则表示需要对测量临时表进行检查，主要是判断是否存在重复数据（也就是在某一个时间段内，某一个网元是否有超过一条的性能数据），如果存在重复数据的话，则把重复数据从测量临时表中删除；设为 0，则不作检查。为了提高入库效率，通常设为 0，即不作检查。
Stepactive	采集动作。用于定义测量临时表到临时主表的入库方式，包括 INSERT 和 UPDATE 两种方式。
Measurement	测量名，也就是测量临时表的名字，在 MEASUREMENT 小节 msnt 进行定义
WHEREGROUP	从测量临时表中选取数据的条件。

5） [性能采集名_STEP_BEFORE]

如果[性能采集名_STEP]小节中某一个采集步骤对应的StepBefore设置为1，则需要在本段定义执行的SQL语句。

```
[TPM_MSC_NOKIA_STEP_BEFORE]
#=============================================================
#!Step          SQLStatement
#=============================================================
Step001 insert into [MSNT:p_msc_all] (msc_rdn,scan_start_time,scan_stop_time) select distinct msc_rdn,scan_start_time,scan_stop_time from [MSNT:p_msc_vlr] where not exists (select msc_rdn,scan_start_time,scan_stop_time from [MSNT:p_msc_all] where [MSNT:p_msc_all].msc_rdn=[MSNT:p_msc_vlr].msc_rdn and [MSNT:p_msc_all].scan_start_time=[MSNT:p_msc_vlr].scan_start_time and [MSNT:p_msc_all].scan_stop_time=[MSNT:p_msc_vlr].scan_stop_time)
#=============================================================
```

列名	含义
Step	采集步骤名。在[性能采集名_STEP]进行定义
SQLStatement	SQL语句

6） [性能采集名]

该段用于定义测量临时表和临时主表的对应关系，以及临时主表和性能目标表的对应关系。

```
[TPM_MSC_NOKIA]
#=============================================================
#!Step                    column          type      arithmetic
#=============================================================
Step001|tpd_msc|temptable|rdn|unique          object_rdn          varchar(40)
```

```
    msc_rdn
Step001|tpd_msc|temptable|starttime|unique      scan_start_time
    SQL_TIMESTAMP     scan_start_time
Step001|tpd_msc|temptable|stoptime|unique      scan_stop_time
    SQL_TIMESTAMP     scan_stop_time
Step001|tpd_msc|temptable          old_rdn          varchar(40)   old_rdn
Step01|tpd_msc|temptable        vlr_roam_sub        SQL_INTEGER
    avg(succ_dep_visits-succ_arr_visits)
#=================================================================
```

列名	含义
Step	采集步骤名。在[性能采集名_STEP]节进行定义。
column	临时主表的字段名（该行的 Step 列使用了 temptable 标志）、性能目标表的字段名（该行的 Step 列使用了性能目标表的名字标志，比如对 MSC 作采集的时候，使用了 tpd_msc 作为标识）
type	字段对应的数据类型
arithmetic	该字段对应的算法。这里的算法是由测量临时表的字段构成，也可由常量构成，也可是 SELECT 语句中的聚合函数。

4．3　工作量总结

4．3．1　LDAP 配置

每添加一个设备，需要在 LDAP 目录添加一系列节点。

LDAP 节点类型	节点名称	备注
Datacheck 类型	d401_msc	针对底层表 tpd_msc
	d401_msc_ho	针对底层表 tpd_msc_ho
	d401_bsc	针对底层表 tpd_radio_bsc
	d401_hlr	针对底层表 tpd_hrl
	d401_bts	针对底层表 tpd_radio_bts
	d401_adj	针对底层表 tpd_adjacent

	d401_tkgp	针对底层表 tpd_tkgp
	d401_link	针对底层表 tpd_link
	d401_destcode	针对底层表 tpd_destcode
	d401_clearcode	针对底层表 tpd_clear_code
OMC 类型	401	标识为 401 的 OMC 配置信息
DSN 类型	401	数据库模式采集数据源定义
	401_txt	文件模式采集数据源定义

4.3.2 Invoker 配置

每开发一个新版本设备，需要在 Invoker 配置文件 GP-NPM_invoker.xml 中添加相应的 XML 格式文本段，实现 Invoker 调用。

4.3.3 DataCheck 厂家模块

DataCheck 厂家模块由程序文件和 xml 配置文件共同组成，同一个版本的设备，拥有同一厂家模块。

DataCheck 模块	文件名	备注
程序文件	nokia_t11_pm_datacheck.pl	
配置文件	nokia_msc.xml	针对底层表 tpd_msc
	nokia_msc_ho.xml	针对底层表 tpd_msc_ho
	nokia_bsc.xml	针对底层表 tpd_radio_bsc
	nokia_hlr.xml	针对底层表 tpd_hrl
	nokia_bts.xml	针对底层表 tpd_radio_bts
	nokia_adj.xml	针对底层表 tpd_adjacent
	nokia_tkgp.xml	针对底层表 tpd_tkgp
	nokia_link.xml	针对底层表 tpd_link
	nokia_destcode.xml	针对底层表 tpd_destcode
	nokia_clearcode.xml	针对底层表 tpd_clear_code

4．3．4 文件获取及解析模块

对于受设备异构性影响最大的文件获取（File_collector）模块与文件解析（Parser）模块，通常由开发人员根据设备商提供的设备信息以及算法映射文档设计程序。根据本人对 NOKIA NMS2000 T12 版本网络设备的分析，设计如下：

（1） 需要在 NOKIA OMC 数据库上为网管创建用户，该用户具有访问数据库的权限，网管系统可以通过远程 rsh 或 telnet 的方式连接到 OMC 上，调用 NOKIA 主机提供的 exemmlmx 工具，以便发指令取文件。

（2）在 Nokia OMC 主机允许目录下放置少量程序以及配置文件辅助采集过程，所有程序以及配置文件，都是为了配合网管系统的采集调用，不增加 NOKIA OMC 主机的资源开销。获得的文件在对方主机的特定目录下生成。

（3）采集服务器主机程序 Nokia_t12_s_getdata.pl 和 Nokia_t12_pm_catfiles.pl，实现文件获取和解析，将文件采集的数据指标以指定格式存放于采集服务器文本数据库中，存储格式以文件 QETXT.INI 定义。为了不让文本数据库中的数据无限量增大，作者设计保留最近五天数据。

4．3．5 Mapping 算法映射文件

作为 ODBC_collector 和 Adaptor 模块共用的 Mapping 算法映射文件是根据设备版本创建的，一个版本设备一个 Mapping 文件，全路径文件名在 LDAP 目录 OMC 类型节点参数中指定。需要对厂商设备主机数据库、采集服务器文本数据库、以及采集服务器数据库有详细了解，并根据业务需要在数据源间建立正确合理的映射及算法关系，实现正确快速的数据库访问。

4．4 本章小节

不同运营商、不同省的移动通信网会使用相同或者不同的通信网络设备。在开发上，采集项目组开发人员根据设备厂商分类进行

采集开发。对设备按照厂商、设备版本制作独立的安装包，比如以上实例，制作 NOKIA NMS2000 T12 设备采集安装包，并且记录制作安装包的时间，以区分同一版本设备的开发升级情况。当省级项目数据采集系统进行施工时，先安装采集核心系统，再按照现场已有的设备情况，使用相应设备的采集程序安装包调试安装。

采用“采集核心系统＋设备采集程序安装包”的方式开发，核心系统与具体的通信设备厂商和版本无关，保证了核心控制系统的稳定性。同时，针对具体设备进行采集开发，保证了采集系统的可扩展性，施工时，只要根据不同项目因时因地制宜，灵活安装，就可以构建完整系统。

5 采集系统进一步要解决的问题

采集系统也是不断升级、不断需要改进的开发与施工过程，下面列出几个正在进行的升级方向。

5．1 实现算法映射文件本地化

ODBC_collector 和 Adaptor 两个模块，解析 Mapping 文件，实现数据的入库算法。Mapping 文件的区分原则是厂家、设备及其版本，在实际投入运行的过程中，同样的设备，用户对算法还有其它方面的要求，需要对算法进行一定增加和修改。

以前的做法是直接将某设备通用的 Mapping 文件进行现场改动，开发工程师按照标准算法文档开发映射文件，现场支持工程师又常会根据现场情况修改算法，在二期系统双方都在一个算法文件上修改，势必影响系统实际应用，而且无法对算法版本进行有效控制。所以，目前着手开发“通用 Mapping+本地 Mapping”结合的方法。开发工程师开发通用标准的 Mapping 文件，现场工程师按照一定规范开发一个现场的本地 Mapping 文件，ODBC_collector 和 Adaptor 执行程序时，综合考虑两个 Mapping 文件，如果本地算法相对于通用算法有增加或者相同的部分，则取本地算法。

实现两个算法文件结合的方式，一来有利于开发人员进行通用算法的版本控制、程序升级，二来有利于现场技术支持工程师为适应现场需求增改部分算法。既保持了版本的一致性，同时又具备了现场灵活性。

5．2 与其它采集系统的融合问题

目前国外也有支持一些厂商及其设备数据采集的较成熟的系统，比如新引进的 ADC 公司的数据采集系统。如何将 ADC 系统和本公司开发的网管数据采集系统做到很好的融合，实现优势互补，是正在着手思考的问题。同时，将来有可能会面对更多有关系统整合的问题。

5．3　文件获取模块整合规划

在 Invoker 调用不同模块实现采集的过程中，File_collector（文件获取），Parser（文件解析）两部分一直是程序员独立设计，每位开发人员针对自己负责的厂商设备，独立作业。文件解析部分由于文件内容、形式不同，需要独立编程，但是文件获取部分对于一部分设备仍有一定规律。采集系统会进一步研究如何开发这些一致性，设计 File_collector 模块。

6 结束语

GSM 网三期网管系统中标移动、联通等电信运营商共 54 省项目，目前，三分之二以上的省级项目已投入运行，并陆续通过用户初验。

电信网网管已经经历了从无到有的过程，现在正在从初级向高级，从不完善向完善，从分散向集中的目标发展。电信网的网络管理是为了促进电信网的运营和发展，所以，无论如何发展变化，电信网网管都应该实现监测网络运行状况，分析网络运行指标，提高网络服务质量。

电信网络的管理内容、管理方式是变化发展的，因此，电信网网管的建设是持续、长期的，电信网网管的建设应该注意可扩展性，保证在网络的规模扩大、网络的业务增长、网络的管理变动时，网管系统在较少的调整下仍然可以适用。电信网出现的业务种类越来越多，数据量越来越大，采集系统也需要不断的改善甚至变革才能满足电信业日益发展的需要。

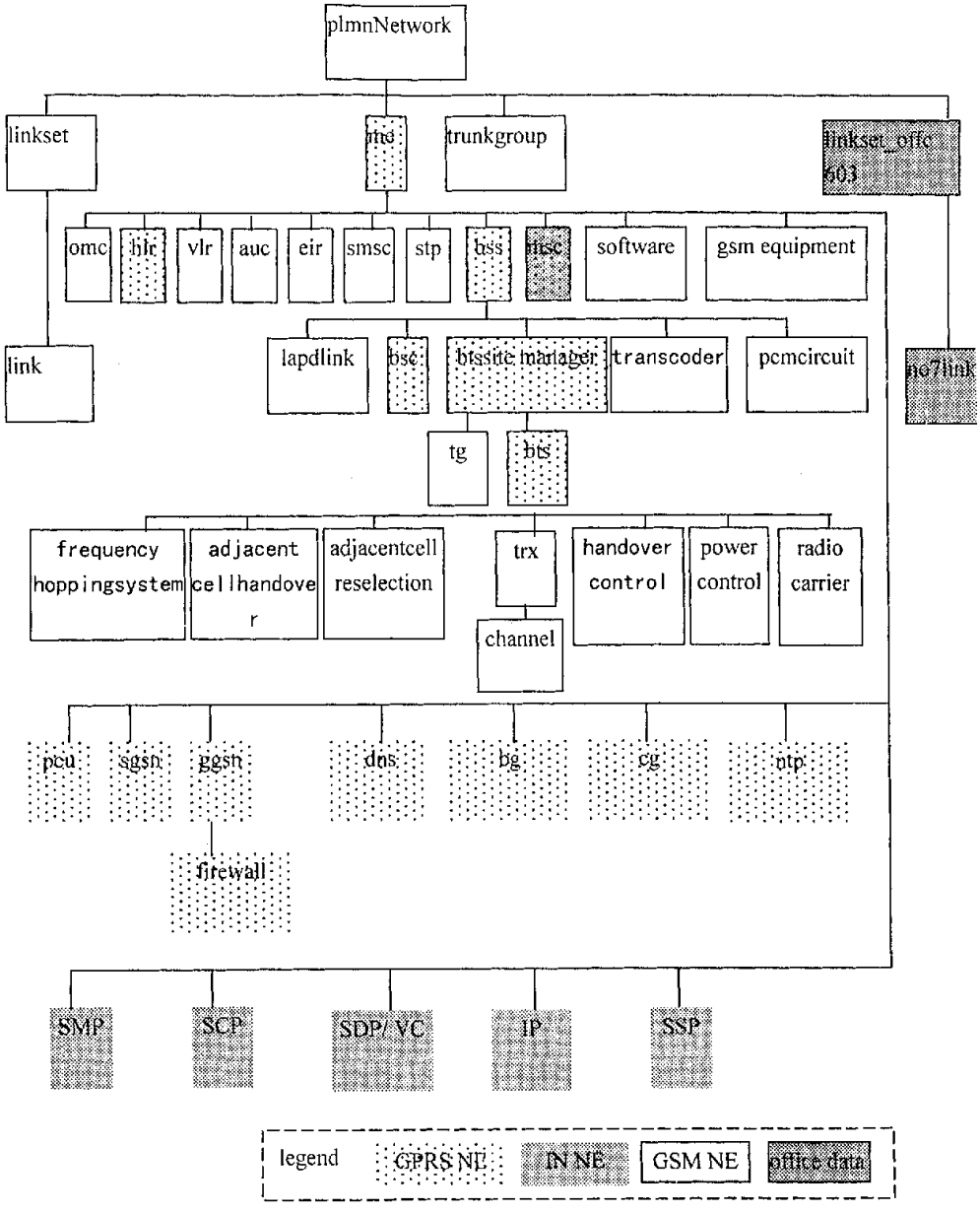
附件
缩略语

缩略语	英文全称	中文译名	说明/定义
TMN	Telecommunications Management Network Model	电信管理网络模型	电信管理网络模型
GSM	Global System For Mobile Communication	全球移动通信系统	第二代移动通信标准之一
LDAP	Lightweight Directory Access Protocol	轻型目录访问协议	轻型目录访问协议
OMC	Operation & Maintenance Center	操作维护中心	操作维护中心
ODBC	Open Database Connectivity	开放式数据库接口	开放式数据库接口
MDDB	MD Database	采集服务器数据库	采集服务器数据库
NPM	Net Performance Management	网络性能管理	网络性能管理
NRM	Net Resource Management	网络资源管理	网络资源管理
MS	Mobile Station	移动台	移动终端
BSS	Base Station System	基站子系统	基站子系统
BSC	Base Station Controller	基站控制器	基站控制器
BTS	Base Transceiver Station	基站收发信台	基站收发信台
NSS	Network Switching System	网络交换子系统	网络交换子系统
OSS	Operation Support System	操作支持子系统	操作支持子系统
MSC	Mobile-service Switching Center	移动业务交换中心	移动业务交换中心
VLR	Visitor Location Register	来访用户位置寄存器	来访用户位置寄存器

HLR	Home Locate Register	归属用户位置寄存器	归属用户位置寄存器
AUC	Authentication Center.	鉴权中心	鉴权中心
NMC	Network Management Center	网络管理中心	网络管理中心
EIR	Equipment Identification Register	移动设备识别寄存器	移动设备识别寄存器
GUI	Graphical User Interface	图形用户界面	图形用户界面
C/S	Client/Server	客户机/服务器	客户机/服务器
B/S	Browser/Server	浏览器/服务器	浏览器/服务器
CM	Configuration Management	配置管理	配置管理
PM	Performance Management	性能管理	性能管理
NRM	Network Resource Management	网络资源管理	采集配置管理数据
NPM	Network Performance Management	网络性能管理	采集性能管理数据
LDAP	Lightweight Directory Access Protocol	轻型目录访问协议	一种目录型数据库
ODBC	Open Database Connectivity	开放数据库互连	是应用程序和数据库系统之间的中间件

网元关系图

网元间的信息模型关系如下所示。


plmnNetwork
linkset
me
trunkgroup
linkset_offc603
omc
hlr
vlr
auc
eir
smsc
stp
bss
msc
software
gsm equipment
link
lapdlink
bsc
btssite manager
transcoder
pcmcircuit
no7link
tg
bts
frequency hoppingsystem
adjacent cellhandover
adjacentcell reselection
trx
channel
handover control
power control
radio carrier
pcu
sgsn
ggsn
dns
bg
cg
otp
firewall
SMP
SCP
SDP/ VC
IP
SSP
legend
GPRS NE
IN NE
GSM NE
office data

参考文献

[1] 孙儒石，丁怀元，穆万里，王泽权编著,中国通信学会主编. GSM 数字移通信工程. 北京:人民邮电出版社，2002 年 5 月

[2] 邵世祥，戴美泰,林纲，吴志忠等编著，中国通信学会主编. GSM 移动通信网络优化. 北京:人民邮电出版社，2003 年 4 月

[3] Larry Wall, Tom Christiansen & Jon Orwant 著, 何伟平 译. Perl 语言编程（第 3 版）. 北京：中国电力出版社，2001 年 12 月

[4] (美)Descartes,A.著，张家万译. Perl DBI 编程. 北京：中国电力出版社，2001 年 2 月

[5]（美）Randal L.Schwartz,Tom Phoenix 著，李晓峰译. PERL 语言入门. 北京：中国电力出版社，2002 年 8 月

[6] 美 Jasnowski,M.著，盖江南等译 .Java,XML 和 Web 服务宝典 . 北京:电子工业出版社,2002 年 5 月

[7] Abraham Silberschatz 著，杨冬青 唐世渭 等译. 数据库系统概念. 北京:机械工业出版社,1999 年 3 月

[8] （美）Eric J.Naiburg,Robert A.Maksimchuk 著，陈立军 郭旭译 .UML 数据库设计应用.北京:人民邮电出版社，2002 年 11 月

[9] CPAN Documentations, Net::LDAP module documentation , http://search.cpan.org/~gbarr/perl-ldap/lib/Net/LDAP.pod

[10] CPAN Documentations, Net::LDAP::FAQ, http://search.cpan.org/~gbarr/perl-ldap/lib/Net/LDAP/FAQ.pod

[11] CPAN Documentations, ML::Parser module documentation, ttp://search.cpan.org/~msergeant/XML-Parser/Parser.pm

[12] Sun System Technical Articles, Text Databases on UNIX or Linux, ttp://developers.sun.com/asp/kb/kb_display_200110246.html

[13] Jonathan Eisenzopf, Perl XML FAQ, http://xml.coverpages.org/perl-xml-faq11.html

[14] CPAN Documentations, ML::Simple module documentation, ttp://search.cpan.org/~grantm/XML-Simple-2.14/lib/XML/Simple.pm

[15] DataDirect Documentations, ODBC Driver - Product Overview, http://www.datadirect.com/products/odbc/odbcoverview/index.ssp

[16] DataDirect Documentations, What is a Wire Protocol ODBC Driver?, http://www.datadirect.com/products/odbc/odbcwireprot/index.ssp

[17] 中国移动文档,《中国移动省级话务网管性能、配置接口方案一NOKIA分册 OSS3.1.doc》

[18] 中国移动文档, 《中国移动话务网网管系统二期改造方案-Nokia 接口分册(V1.5.2).doc》

[19] 丁 丽, 中国网络通信有限公司网络管理部, 综合网管建设探讨, http://www.cww.net.cn/Technique/2004/12/19286.htm, 2004 年 12 月

[20] 亿阳信通网管培训文档, 移动省级话务网三期网管系统培训教程,2004年

[21] DataDirect Documentations , H The UNIX Environments , http://media.datadirect.com/download/docs/odbc/odbcref/unixenviron.html

[22] David A.Curry 著, 孙伟峰译. UNIX 系统编程——基于 SVR4. 北京:中国电力出版社, 2001 年 7 月

[23] Kay A Robbins 著, 刘宗田等译. 实用 UNIX 编程. 北京: 机械工业出版社, 1999 年 9 月

[24] 亿阳公司产品文档, 传输综合网管系统 , http://www.boco.com.cn/member/xtchanpin/wangguan2.htm

[25] 亿阳信通网管系统安装文档, MC-NMS-V4.0_P_SIM_iPlanetDirectoryServer_031219.doc

[26] 亿阳信通网管系统安装文档, MC-NMS-V4.0-NMOS_P_SIM_UNIX 软件环境安装配置_040219.doc

[27] David Tansley 著, 张春萌等译. LINUX 与 UNIX SHELL 编程指南. 北京: 机械工业出版社,2000 年 6 月

[28] 亿阳信通培训文档, 电信管理网 (TMN) 简介

[29] 亿阳信通培训文档,GSM 移动通信系统介绍

[30]（美）Ron Flannery 著，邱仲潘 等译. INFORMIX 实用全书. 北京：电子工业出版社，2002 年 1 月

[31] George Reese Randy Jay Yarger Tim King 著，林琪、朱涛江译.MySQL 权威指南（第二版）.北京：中国电力出版社，2003 年 5 月

[32] MYSQL Documentations，http://dev.mysql.com/doc/

[33] Carol McCullough-Dieter 著，天宏工作室译. ORACLE 9I 数据库管理员——实现与管理. 北京：清华大学出版社，2004 年 2 月

作者在读期间科研成果简介

1） 亿阳信通四川分公司，参与四川移动 EOMS v1.5 期系统开发。

2） 亿阳信通四川分公司，参与四川移动话务网管二期系统二次开发。

3） 亿阳信通四川分公司，参与四川移动 GSM 网络预警系统开发。

4） 亿阳信通四川分公司，参与四川移动告警系统细粒度网元配置数据采集开发。

5） 亿阳信通北京总公司，参与话务网管三期数据采集系统开发工作。

6）亿阳信通云南分公司，参与云南移动二期网管 NOKIA-SGSN 设备采集开发。

7） 发表论文：数据仓库技术在 GSM 网络预警系统中的应用。《信息工程学院学报》2005 年 6 期。

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的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得四川大学或其他教育机构的学位或证书而使用过的材料。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。

本学位论文成果是本人在四川大学读书期间在导师指导下取得的，论文成果归四川大学所有，特此声明。

导师：

学生：

2005 年 4 月

致谢

三年的研究生学习生涯即将结束，老师与同学在学习和生活上对我的关怀和帮助使我受益匪浅。

首先，我要感谢我的导师李志蜀教授，他对待学术作风严谨，对待学生平易近人、和蔼可亲。他鼓励学生积极实践，当我远去北京做开发工作时，他给予了强大的支持，在此，我由衷的表示感谢。相信当我走出校园，迈入社会后，老师的做人原则会继续激励、影响我。

然后，感谢我的师兄师姐，他们从很多方面都给我很好的建议，让我少走弯路；感谢我的同学，大家朝夕相伴，学业和生活上互相帮助、共同进步；还要感谢我的师弟师妹，跟他们交流让我的视野更开阔，大受益处。

在学校期间，有机会去亿阳信通公司实习，使我开阔了眼界。在公司，理论与实践得到了很好的结合，同事的关心、爱护和教导让我又多了一片发展的天空、学习的天地。在此，感谢亿阳信通公司，感谢那里的同事所给我的工作上和生活上的帮助。

最后，我要感谢我的母校四川大学，这里有严谨又不乏开放的学术作风，优美的环境，海纳百川的气势。在四川大学，我长大了，成熟了，是川大培养了我，教导了我。再多的言辞都难以表达我对母校的感激之情，唯有日后更加努力。