移动新媒体平台数据仓库方案系统设计文档

2017.01.19

**深圳市九指天下科技有限公司**

**目录**

[移动新媒体平台数据仓库方案系统设计文档 1](#_Toc496880603)

[1. 编写目的 4](#_Toc496880604)

[1.1 术语与缩略词 4](#_Toc496880605)

[1.2 对象及范围 5](#_Toc496880606)

[1.3 总体设计原则 5](#_Toc496880607)

[1.4 总体技术路线 6](#_Toc496880608)

[1.5 尚未解决的问题 7](#_Toc496880609)

[2. 数据仓库产品需求分析 7](#_Toc496880610)

[3. 数据仓库架构图 8](#_Toc496880611)

[3.1 业务架构图 8](#_Toc496880612)

[3.2 技术架构图 8](#_Toc496880613)

[4. 数据采集 9](#_Toc496880614)

[4.1 数据采集流程图 10](#_Toc496880615)

[4.2 核心概念 10](#_Toc496880616)

[5. 数据存储 13](#_Toc496880617)

[5.1 存储架构 13](#_Toc496880618)

[5.2 核心概念 13](#_Toc496880619)

[6. 数据计算 14](#_Toc496880620)

[6.1 计算平台架构图 14](#_Toc496880621)

[6.2 核心概念 14](#_Toc496880622)

[7. 数据应用 16](#_Toc496880623)

[7.1 用户分析 17](#_Toc496880624)

[7.2 渠道分析 19](#_Toc496880625)

[7.3 终端属性分析 21](#_Toc496880626)

[8. 任务调度与监控告警 22](#_Toc496880627)

[8.1 调度系统架构图 23](#_Toc496880628)

[8.2 任务管理器 23](#_Toc496880629)

[8.3 任务调度器 24](#_Toc496880630)

[9. 参考文献 25](#_Toc496880631)

# 编写目的

此概要设计说明文档的编写目的是为了说明数据仓库系统的体系架构，以及需求用例的各个功能点在架构中的体现，为项目组后续开展详细设计、系统建设与实现、系统测试提供依据。

本概要设计说明书适用于数据仓库详细设计、开发、测试等研究开发工作。

本文档的预期读者包括支持本项目的领导和负责人员；详细设计、开发、测试人员及文档编写人员；项目管理人员；参与项目实施的相关工作人员等。

## 术语与缩略词

|  |  |
| --- | --- |
| **词语** | **解释** |
| DW | Data Warehouse |
| EDW | Enterprise Data Warehouse |
| ODS | Operational Data Store |
| DM | Data Mart |
| BI | Business Intelligence |
| OLAP | On-Line Analytical Processing |
| OLTP | On-Line Transaction Processing |
| Flume | Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data |
| Sqoop | Sqoop is a tool designed to transfer data between Hadoop and relational databases or mainframes |
| Kafka | Kafka is a distributed streaming platform |
| Hadoop | Hadoop is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models |
| HDFS | Hadoop Distributed File System |
| Hive | Hive is the Hadoop data warehouse |
| Hbase | HBase is the Hadoop database, a distributed, scalable, big data store |
| Spark | Spark is a fast and general engine for large-scale data processing |
| Zookeeper | ZooKeeper is an effort to develop and maintain an open-source server which enables highly reliable distributed coordination |
| 业务架构 | 信息化总体架构的起点，业务架构定义了在总体企业战略指导下希望实现的业务能力及其关系。 |
| 应用架构 | 对信息系统进行的高层应用进行划分，指导整个信息系统构建和实施 |
| 数据架构 | 从跨组织应用系统的视角对数据进行组织和管理，包括对整个数据生命周期中数据的处理、存储、转换、整合、分布制定的策略、模型、流程、以及支持这些策略、模型、流程的技术架构方案。 |
| 技术架构 | 基于面向服务架构思路，实现信息、应用、流程横向贯通，支持应用架构优化和提升的技术平台架构；定义各个信息基础设施之间的关系，从宏观和微观角度分析信息系统的发展过程和技术要求，并为保障和支撑应用和数据提供一个可实现的基础。 |

* 1. 对象及范围

程序开发人员、系统架构师。

* 1. 总体设计原则
* 实用性原则

本项目的建设首先要坚持实用性原则，在实用的基础上考虑先进性和前瞻性，选用符合标准的，先进成熟的产品和开发平台，构建一个切合实际、解决实际问题的系统。

* 标准化原则

本项目具有数据量大、应用存有差异的特点，通过典型的应用功能与架构规划，才能为差异化的实施部署提供建设指导。

* 一体化原则

提供统一的功能与服务。

* 适用性原则

充分考虑不同现状，本系统应具备良好的可配置性和可扩展性，满足当前阶段行业现有的需求。

* 可维护性原则

在系统总体设计上注意系统的维护性，尽量采用成熟且易于维护的系统平台，维护应简单、易于操作。

* 安全性原则

系统建设应遵循互联网安全要求。

* 1. 总体技术路线
* 采用多层体系结构，严格按数据仓库和标准java开发规范进行开发。
* 提供灵活的界面展示和良好的扩展性。
* 系统界面简洁规范，美观大方，操作方便；具备人性化、简单的人机交互方式。
* 实现高级别的安全保障系统，充分保证数据和业务的安全性。

表格1 总体技术路线

|  |  |
| --- | --- |
| 分类 | 选型原则 |
| 技术选型 | * 前台界面展现技术采用javascript、Java、JSP；； * 服务器端技术采用j2EE、jdbc、servlet、mybatis,spring等； * Jdk版本1.7，Scala版本2.11，操作系统linux rhel5.7以上 |
| 开发平台 | * SG-UAP |
| Web服务器 | * Tomcat7.0以上版本、ngnix |
| 数据库 | * mysql5.5以上版本 |
| 开源软件 | * Kafka、Flume、Hadoop、Hive、Hbase、Spark、Zookeeper等。 |

* 1. 尚未解决的问题

无

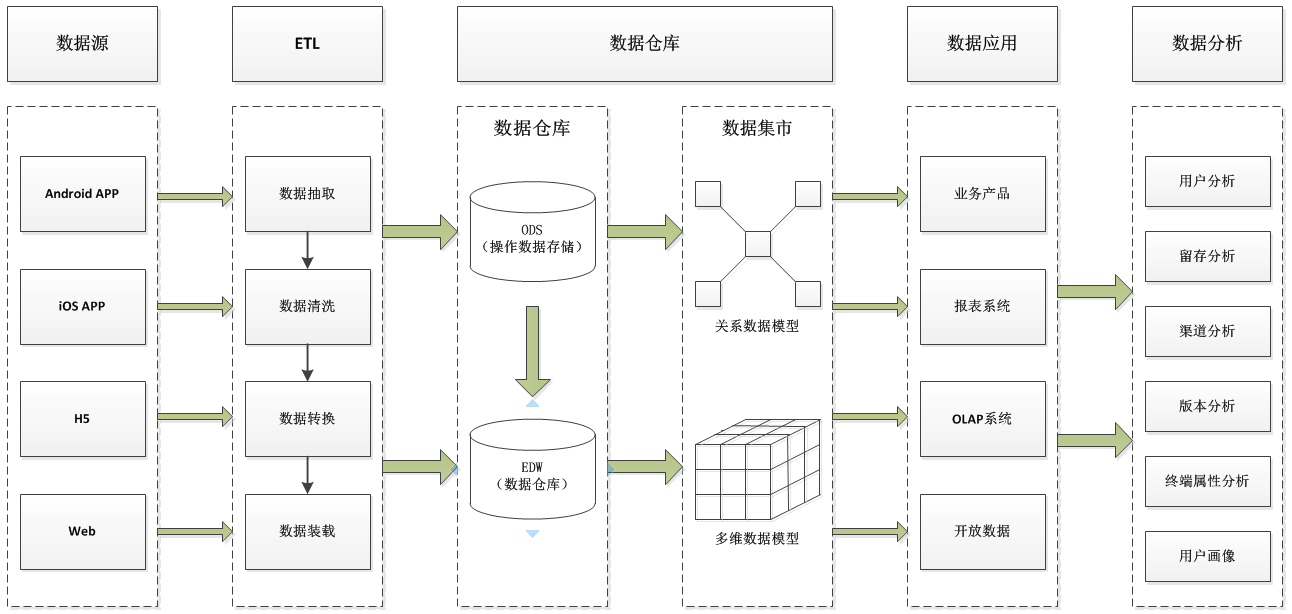
# 数据仓库产品需求分析

为了整合公司所有业务数据，建立统一的数据中心；使企业高层领导能够从全局角度出发，推动企业数据的统一规划，便于业务人员对企业数据的分析与理解；实现统一的报表体系规范，便于实现企业的运营指标分析和统一的口径；形成有效的数据管理体系，保证企业在业务部门众多，内部数据和外部数据复杂的情况下，数据只有唯一事实的特点。必须建设一套完善的数据仓库。

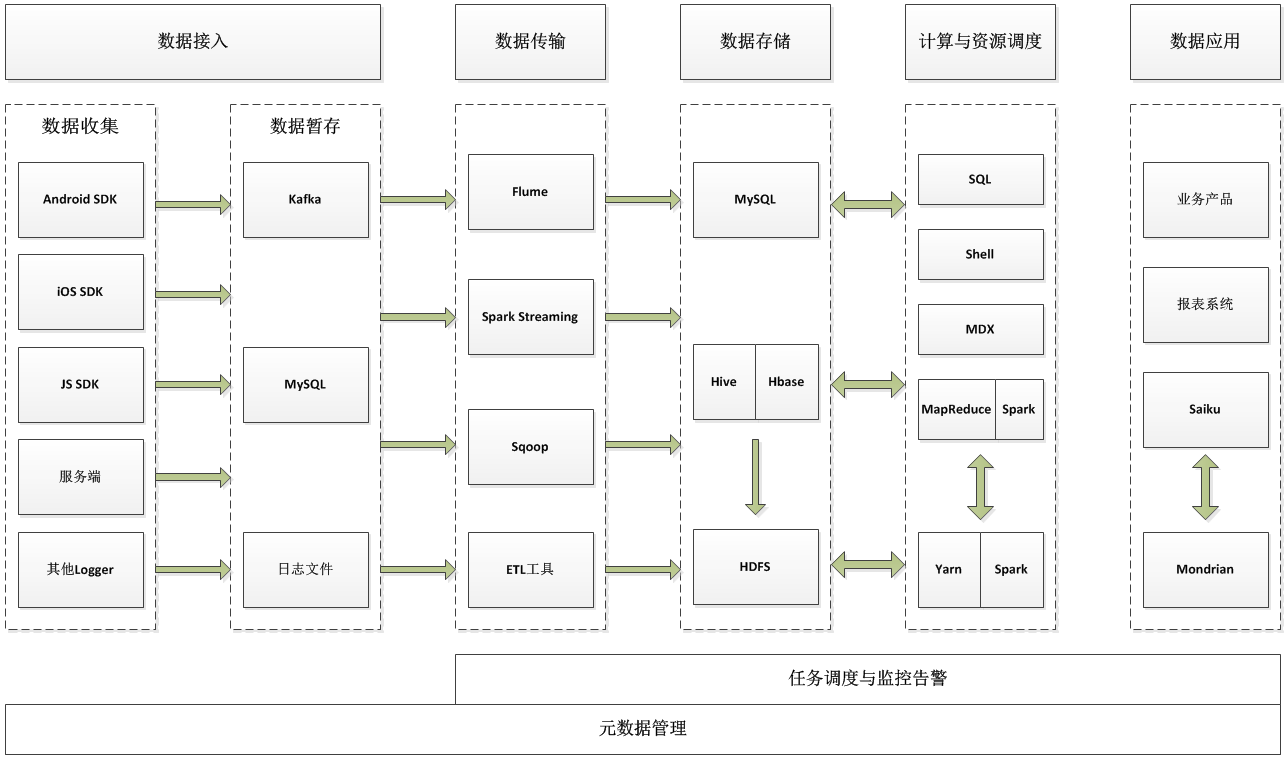
数据仓库必须包含海量数据的收集，分布式存储，分布式并行计算,OLAP分析工具，报表系统等功能特性。

# 数据仓库架构图

## 业务架构图



## 技术架构图



**方案描述：**

**高性能日志收集**

用高性能分布式消息系统kafka作为日志接收和临时存储系统，再用高可靠、分布式日志采集、聚合、传输系统flume将kafka中的日志转存到HDFS。

**分布式存储**

用Hadoop分布式文件系统作为底层存储系统，可以存储海量数据，扩容方便，数据多副本，安全性高，集群高可用。

**并行计算**

用Spark分布式并行计算框架来做数据分析处理，任务并行计算，基于内存计算，效率非常高。

**任务调度与监控告警**

任务调度系统管理任务之间的依赖关系，监控任务的执行状态，任务失败自动重试，邮件、短信实时告警。

**多维分析**

使用Saiku多维分析工具，非常方便地对数据进行任意维度的切片和切块，以快速发现并挖掘数据背后的价值。

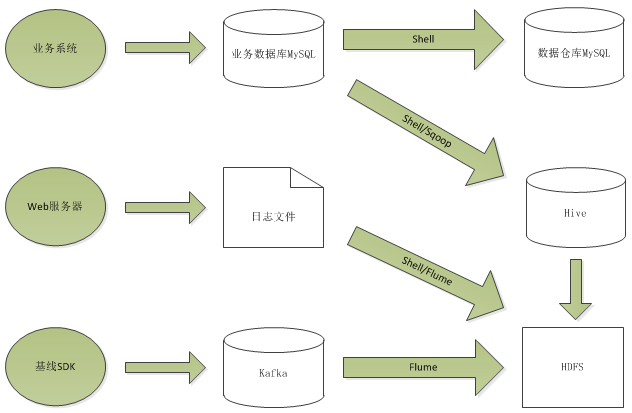
**可扩展性**

存储系统、计算资源等都可以很方便的进行扩展。

# 数据采集

数据采集就是把数据从各种数据源中采集并存储到数据仓库，期间有可能会做一些简单的清洗。目前用到的数据源有MySQL、Kafka、日志文件。

## 数据采集流程图



## 核心概念

**基线SDK**

基线SDK用于收集并上传用户的点击行为日志，可以实时上传或者定时上传。

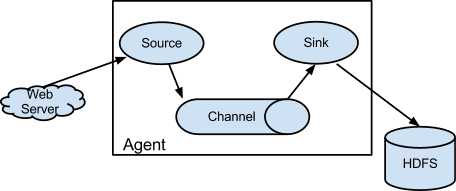
**Kafka**

kafka是一种高吞吐量的分布式发布订阅消息系统。用于接收和临时存储用户点击行为日志。



**Flume**

**flume是cloudera公司开源的一款高性能、高可用的分布式日志收集系统。用于将kafka里的用户行为日志转存到HDFS上。**



**Sqoop**

**sqoop是一个用来将Hadoop和关系型数据库中的数据相互转移的工具，可以将一个关系型数据库（例如 ： MySQL ,Oracle ,Postgres等）中的数据导进到Hadoop的HDFS中，也可以将HDFS的数据导进到关系型数据库中。**

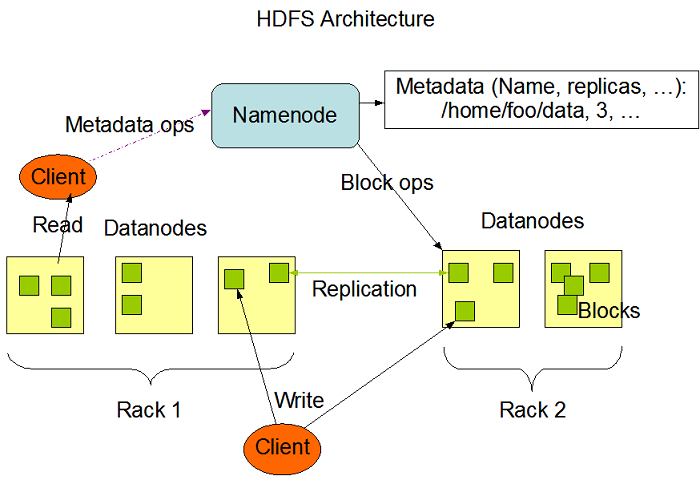
**Hive**

**hive是基于hadoop的一个数据仓库工具，可以将结构化的数据文件映射为一张数据库表，并提供简单的sql查询功能，可以将sql语句转换为MapReduce任务进行运行。 其优点是学习成本低，可以通过类SQL语句快速实现简单的MapReduce统计，不必开发专门的MapReduce应用，十分适合数据仓库的统计分析。**

**HDFS**

HDFS是一个高容错的、高吞吐量的、可运行在普通硬件上的分布式文件系统。

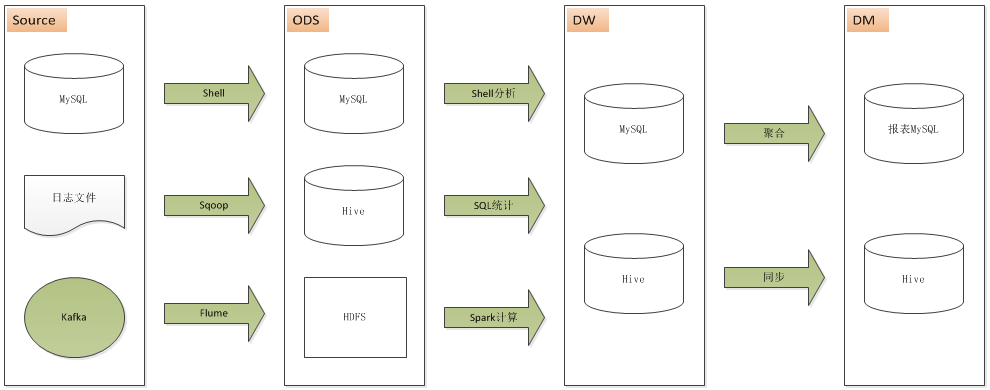
HDFS架构图：

****

# 数据存储

数据仓库存储采用分层架构，自底至上分别为ODS层、DW层、DM层。

## 存储架构



## 核心概念

**ODS（Operational Data Store）**

ODS层是将OLTP数据通过ETL同步到数据仓库来作为数据仓库最基础的数据来源。在这个过程中，数据经过了一定的清洗，比如字段的统一，脏数据的去除等，但是数据的粒度是不会变化的。ODS层的数据可以只保留一定的时间。

**DW（Data Warehouse）**

数据仓库，是为企业所有级别的决策制定过程，提供所有类型数据支持的战略集合。它是单个数据存储，出于分析性报告和决策支持目的而创建。 为需要业务智能的企业，提供指导业务流程改进、监视时间、成本、质量以及控制。

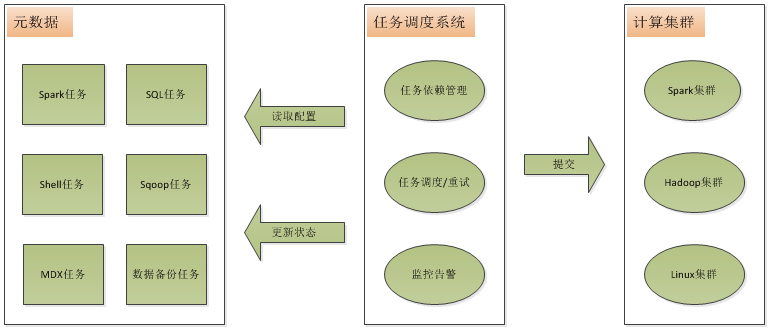
**DM（Data Mart）**

数据集市就是满足特定的部门或者用户的需求，按照多维的方式进行存储，包括定义维度、需要计算的指标、维度的层次等，生成面向决策分析需求的数据立方体。

# 数据计算

计算任务有多种类型，比如spark任务、SQL任务、Shell任务等，每个任务都可以指定计算集群。计算集群按功能可以有执行spark任务的spark集群，执行MapReduce任务的hadoop集群，执行shell脚本任务的linux集群等。

## 计算平台架构图



## 核心概念

**元数据**

**按照传统的定义，元数据（Metadata）是关于数据的数据。在数据仓库系统中，元数据可以帮助数据仓库管理员和数据仓库的开发人员非常方便地找到他们所关心的数据。**

**在数据仓库系统中，元数据机制主要支持以下五类系统管理功能：（１）描述哪些数据在数据仓库中；（２）定义要进入数据仓库中的数据和从数据仓库中产生的数据；（３）记录根据业务事件发生而随之进行的数据抽取工作时间安排；（４）记录并检测系统数据一致性的要求和执行情况；（５）衡量数据质量。**

**任务调度**

**按照任务执行的周期和时间以及任务与任务之间的依赖关系先后启动任务。任务调度系统有以下几个功能：**

1. **周期性的启动任务。**
2. **启动一个任务之前，它所依赖的任务必须执行成功。**
3. **任务失败后自动重试。**
4. **执行任务并行数控制**
5. **监控告警**

**MDX**

**MDX，多维表达式（multi-dimensional expressions）的英文缩写词，是一种编程语言语法，支持多维对象与数据的定义和操作。MDX 在很多方面与结构化查询语言 (SQL) 语法相似，但它不是 SQL 语言的扩展；事实上，MDX 所提供的一些功能也可由 SQL 提供，但是不如MDX有效或直观。**

**Hadoop**

**一个由Apache基金会所开发的分布式系统基础架构。核心组件包括分布式文件系统HDFS和并行计算框架MapReduce。**

**Spark**

**Apache Spark 是专为大规模数据处理而设计的快速通用的计算引擎。Spark是UC Berkeley AMP lab (加州大学伯克利分校的AMP实验室)所开源的类Hadoop MapReduce的通用并行框架，Spark，拥有Hadoop MapReduce所具有的优点；但不同于MapReduce的是Job中间输出结果可以保存在内存中，从而不再需要读写HDFS，因此Spark能更好地适用于数据挖掘与机器学习等需要迭代的MapReduce的算法。**

# 数据应用

**业务系统**

例如排行榜、TopN、用户画像、推荐结果等，用到的数据一般会直接写入业务系统数据库或者是redis缓存。

**OLAP**

**OLAP是在基于数据仓库多维模型的基础上实现的面向分析的各类操作的集合。我们采用了开源的ROLAP引擎Mondrian和前端工具Saiku。**

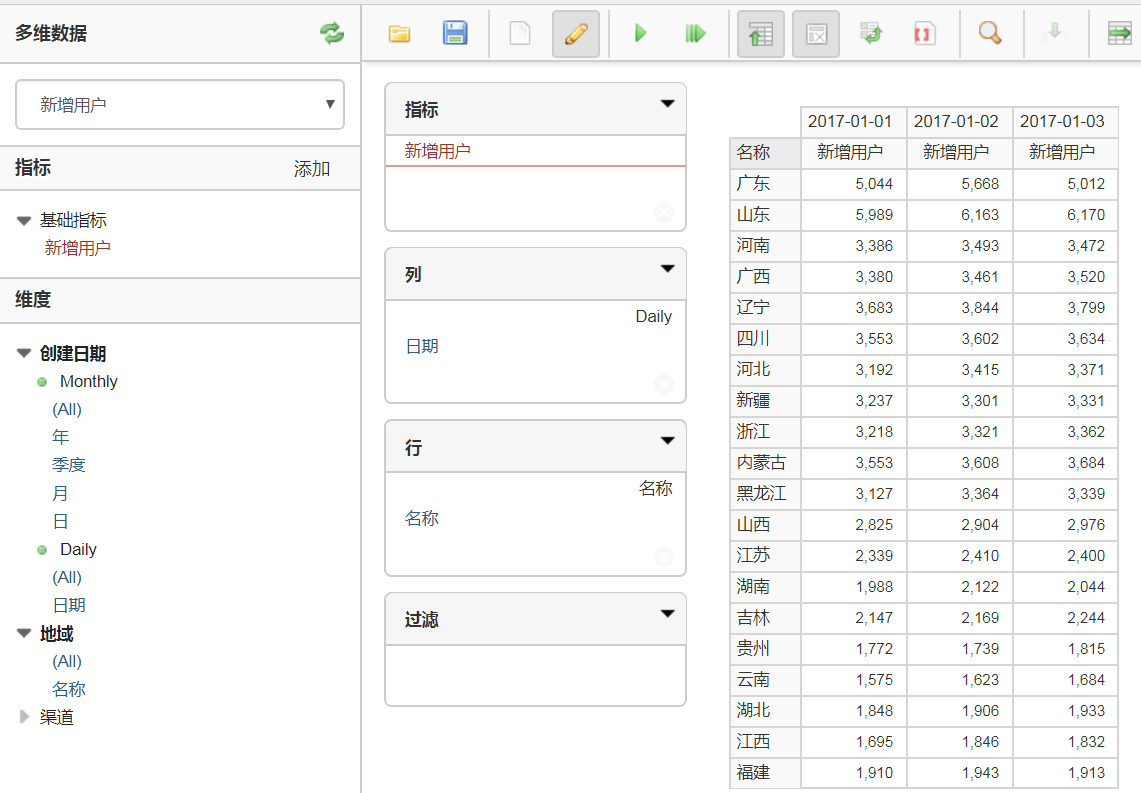
**报表系统**

**一般都是统计汇总好的数据，比如新增用户、活跃用户、留存用户，渠道分析，版本分布等日常运营数据。**

## 用户分析

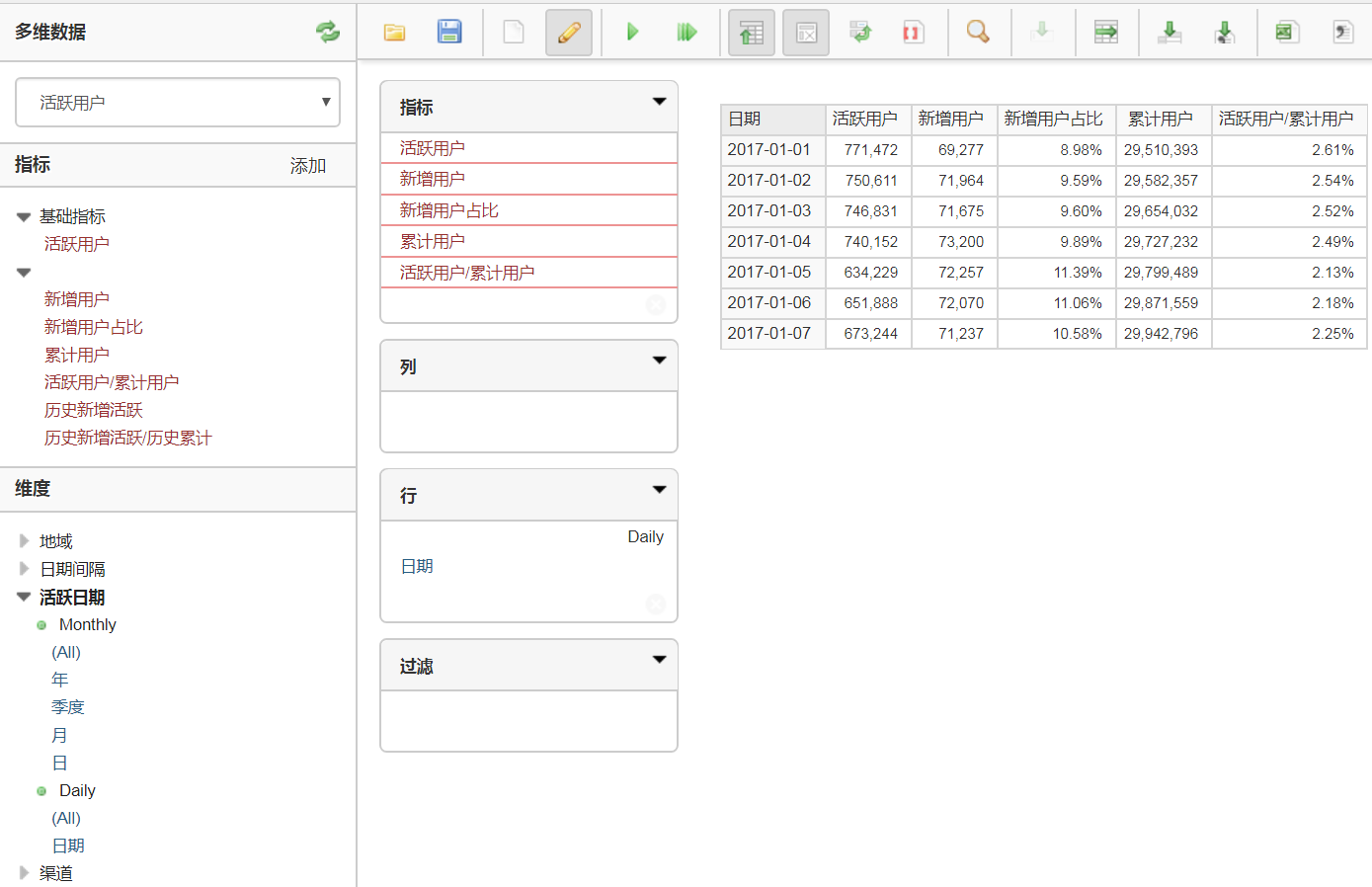
**新增用户**

首次联网启动应用的设备视为一个新增用户，以IMEI/AndroidID/设备ID标识用户的唯一性。如例图所示：



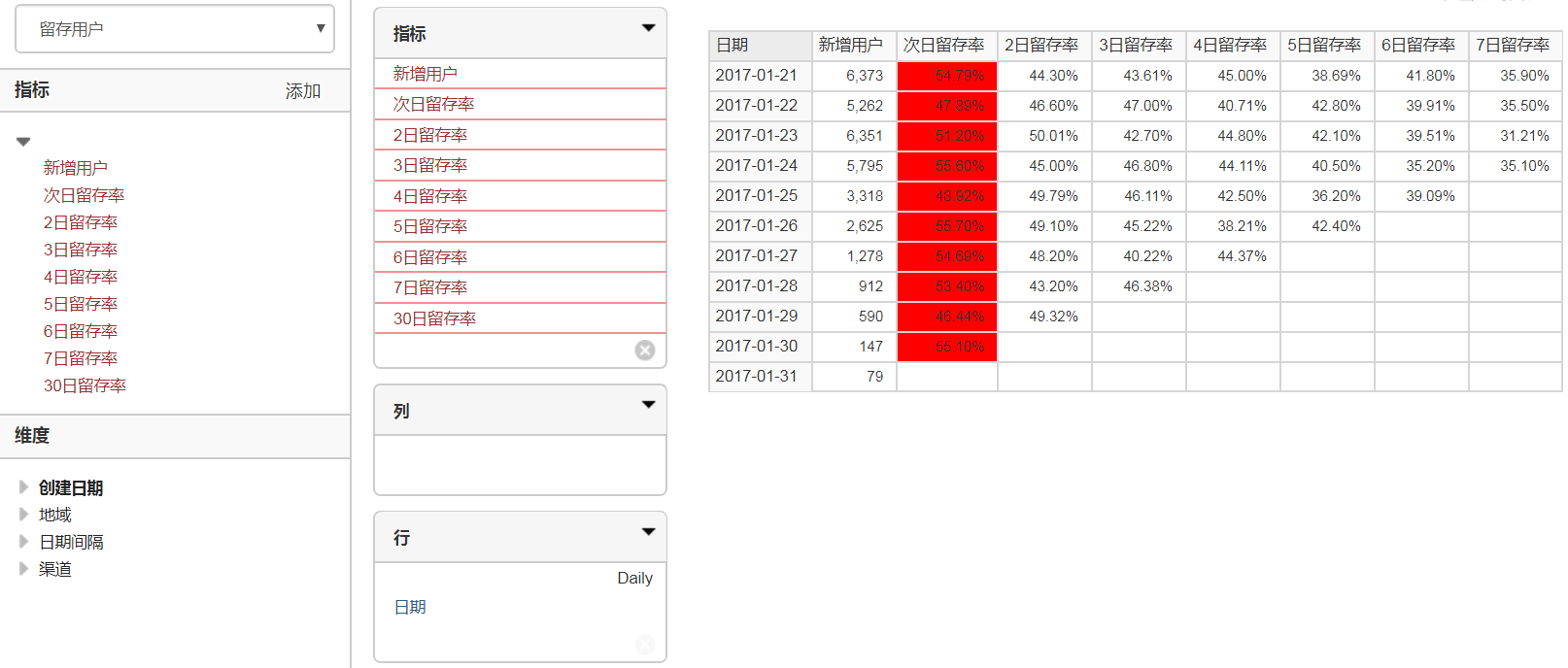
**活跃用户**

所选时间段内，启动过应用的用户(去重)，启动过一次的用户即被视为活跃用户，包括新用户和老用户。如例图所示：



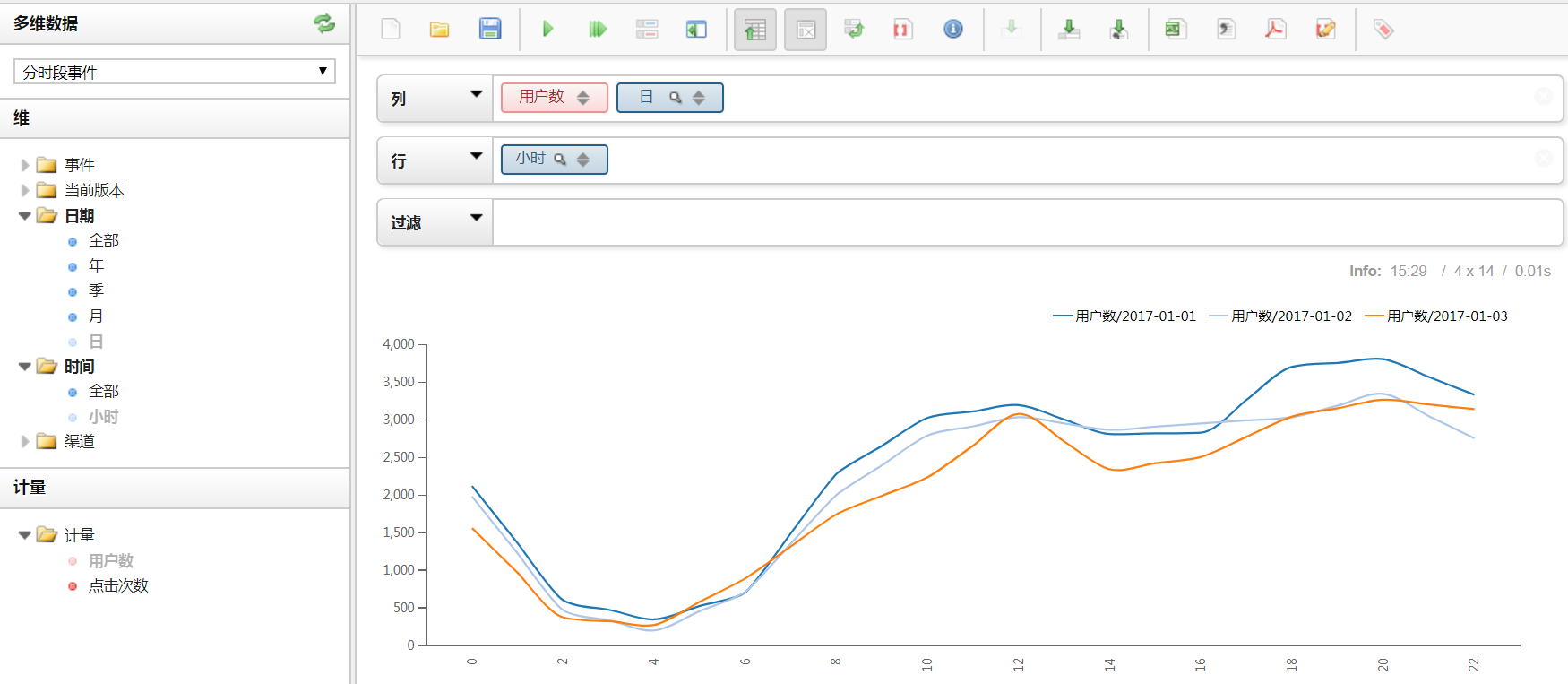
**留存用户**

某段时间内的新增用户，经过一段时间后，仍继续使用应用的被认作是留存用户，这部分用户占当时新增用户的比例即时留存率。如例图所示：



**分时段活跃用户**

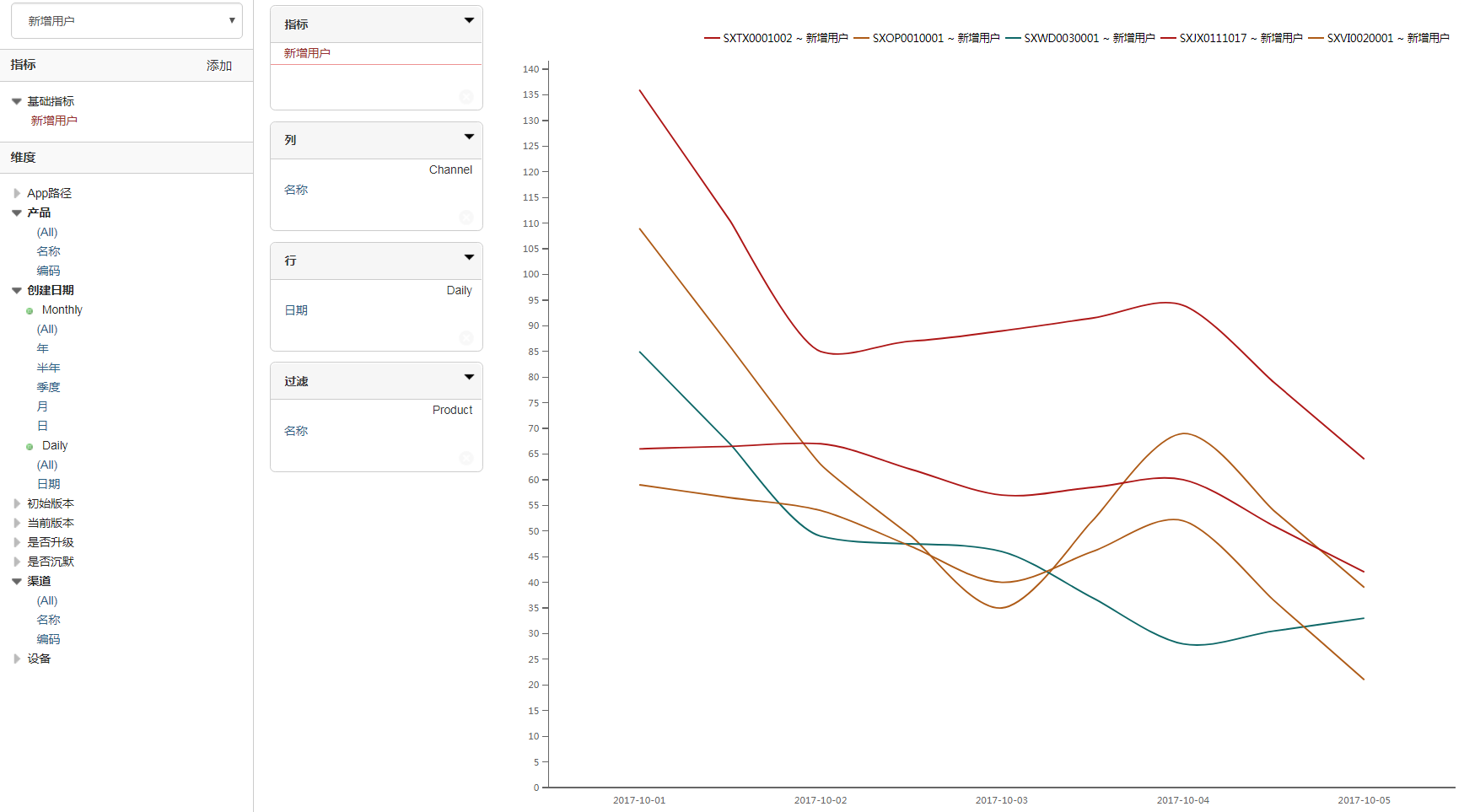
如例图所示：



## 渠道分析

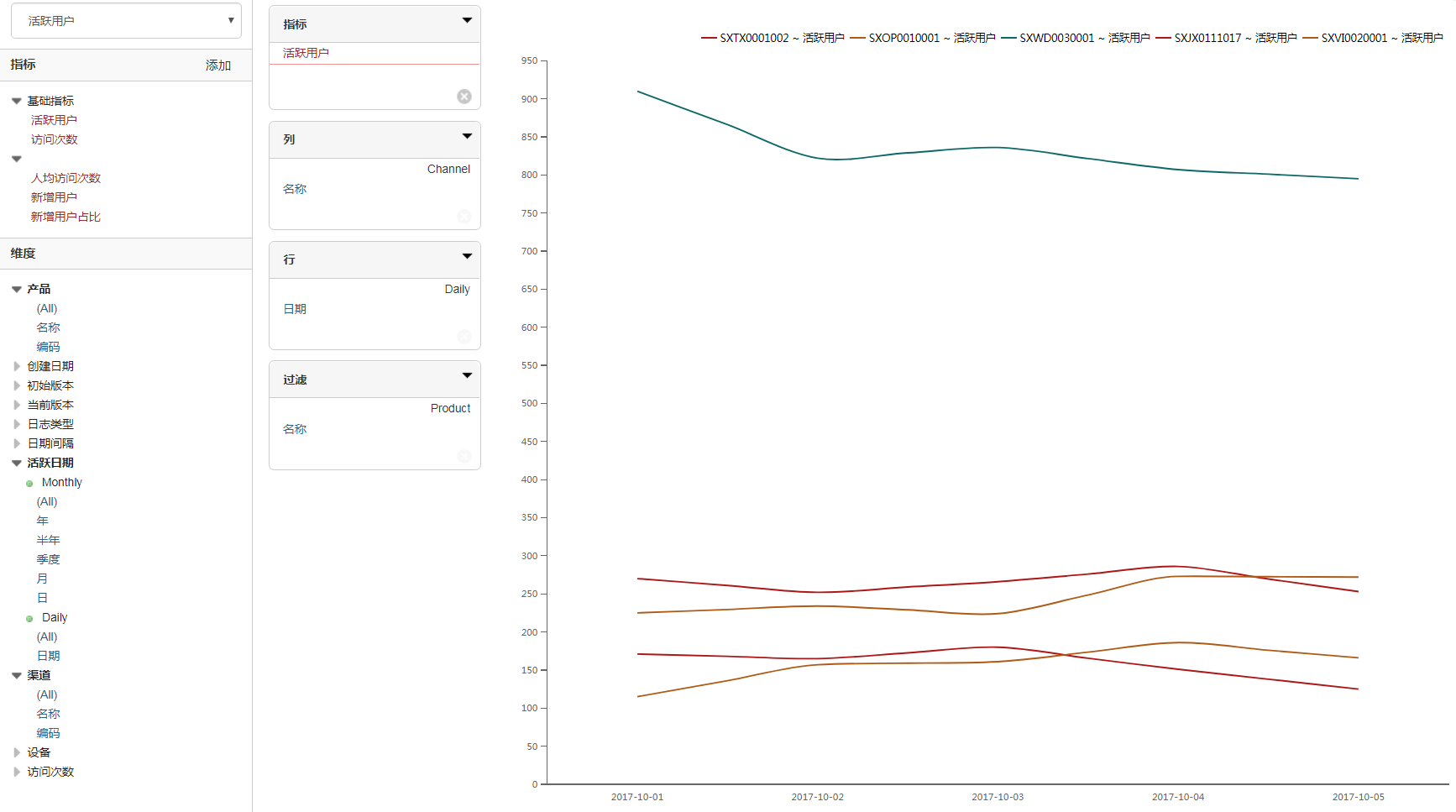
**渠道新增用户**

如例图所示：



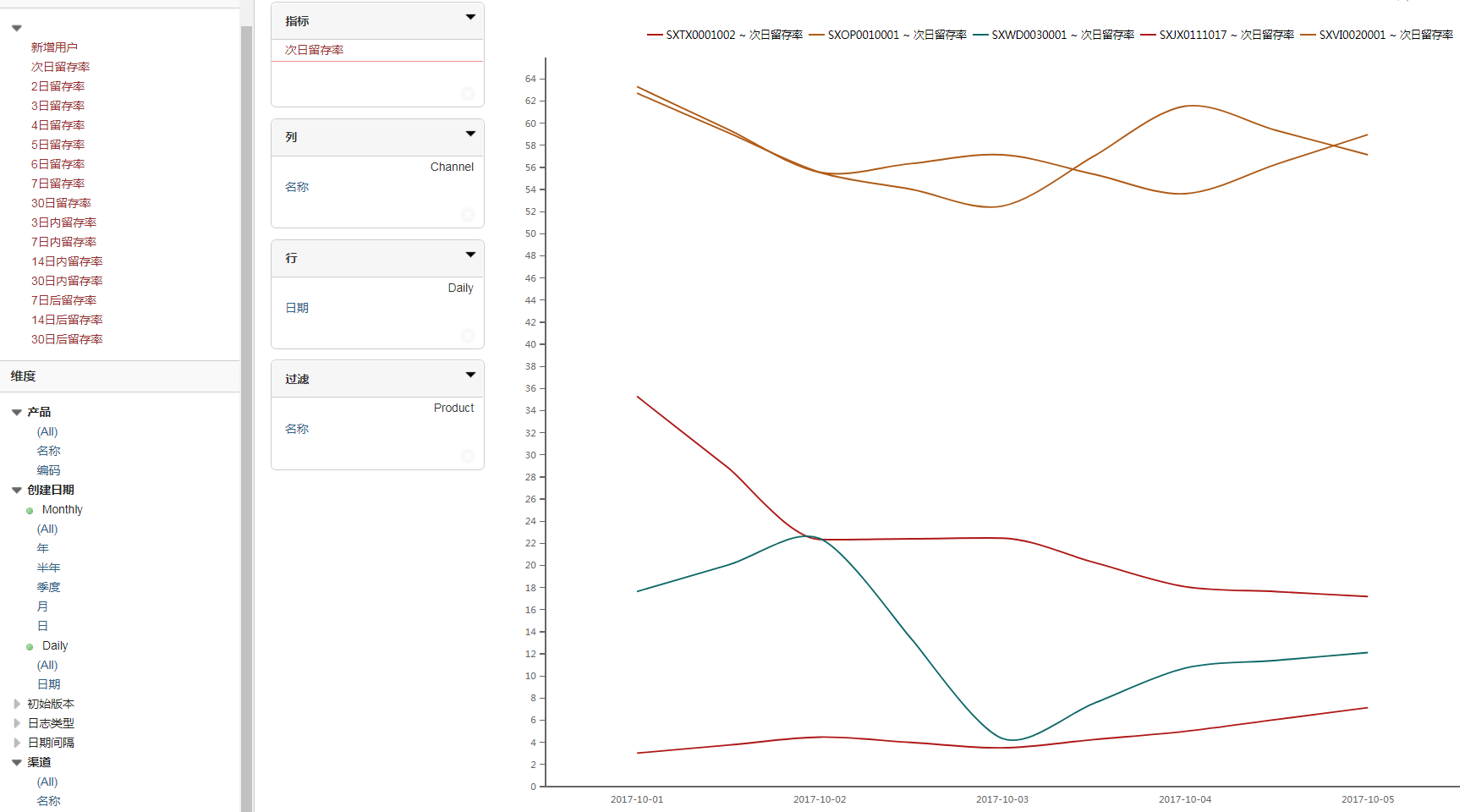
**渠道活跃用户**

如例图所示：



**渠道次日留存率**

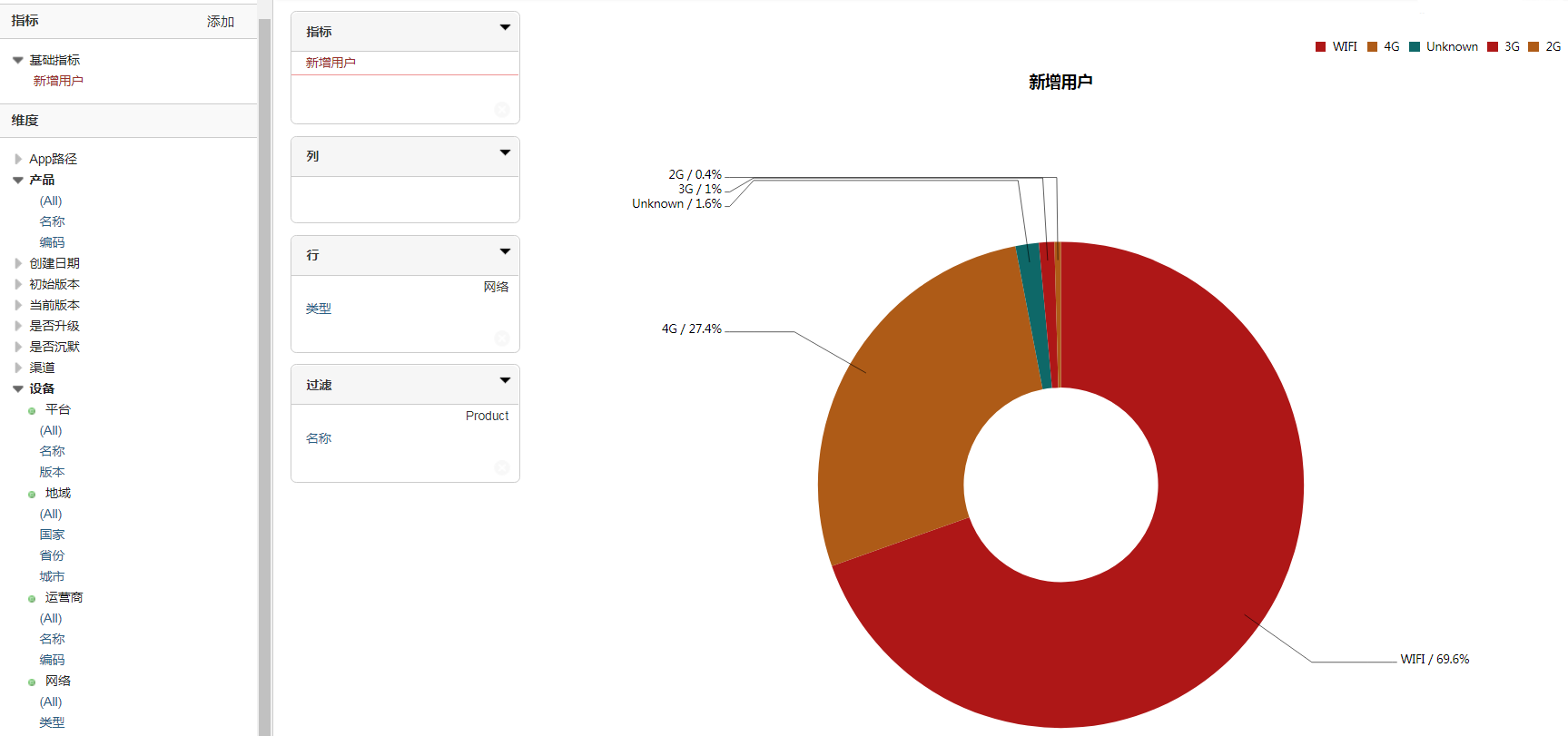
如例图所示：



## 终端属性分析

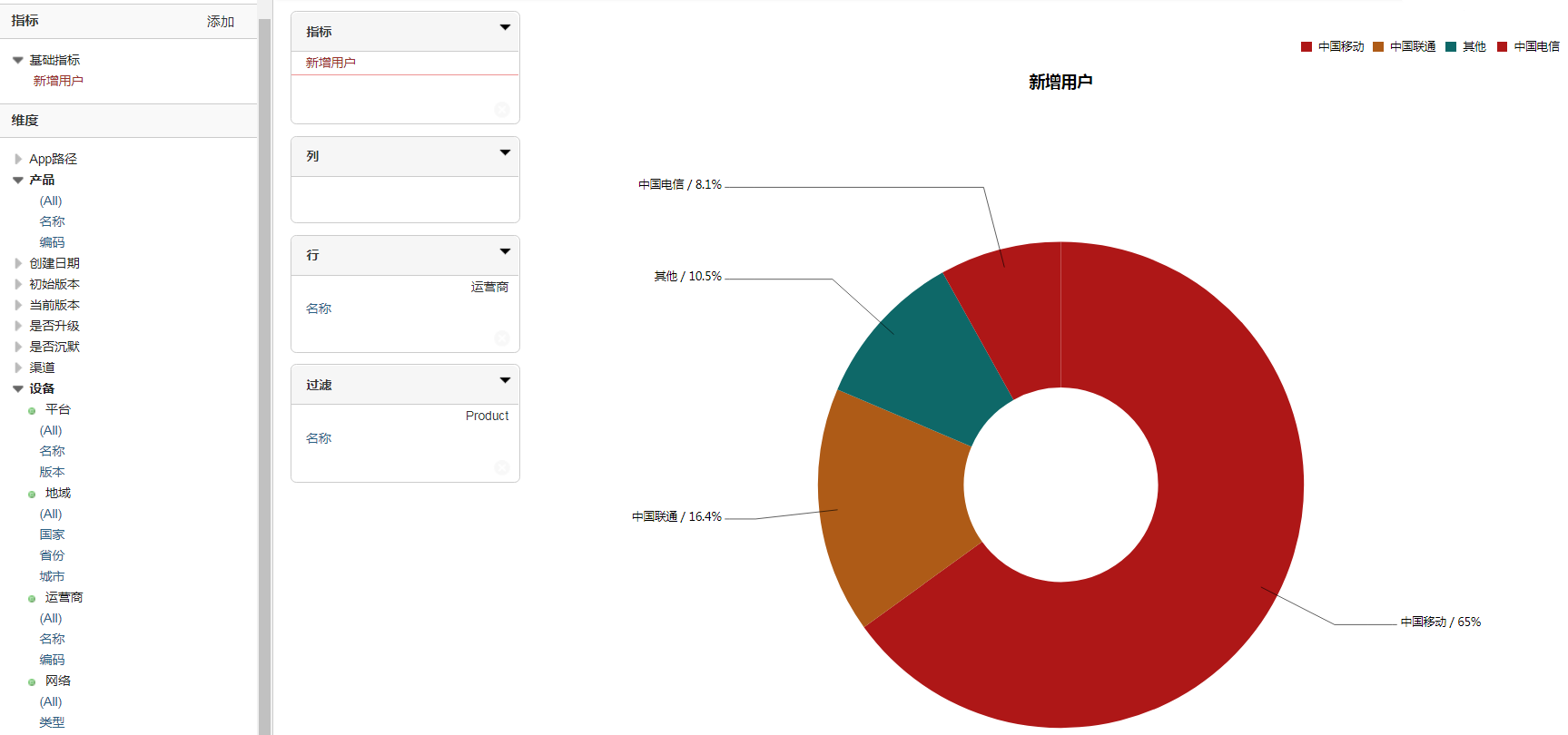
**网络分布**

如例图所示：



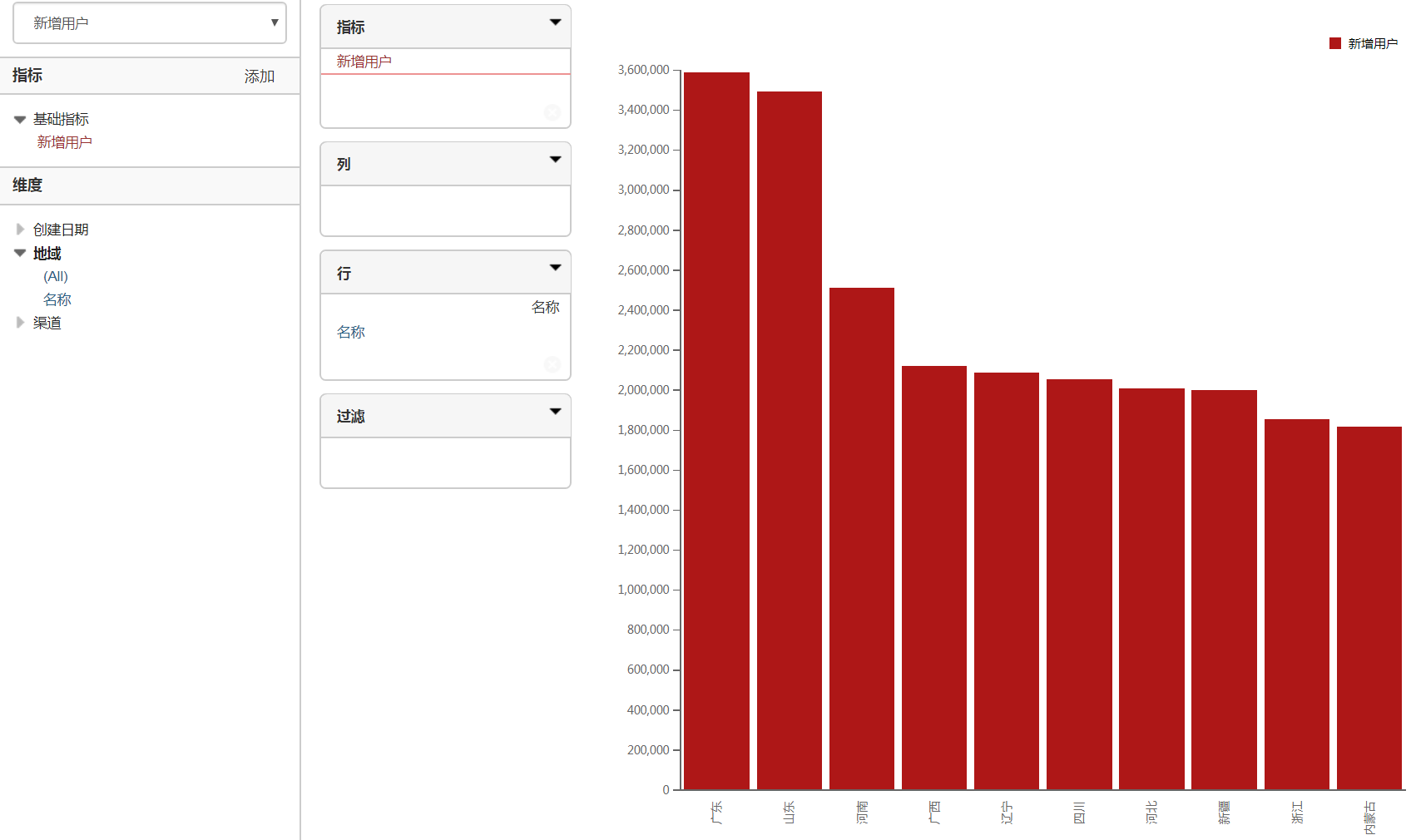
**运营商占比**

如例图所示：



**地区分布**

如例图所示：

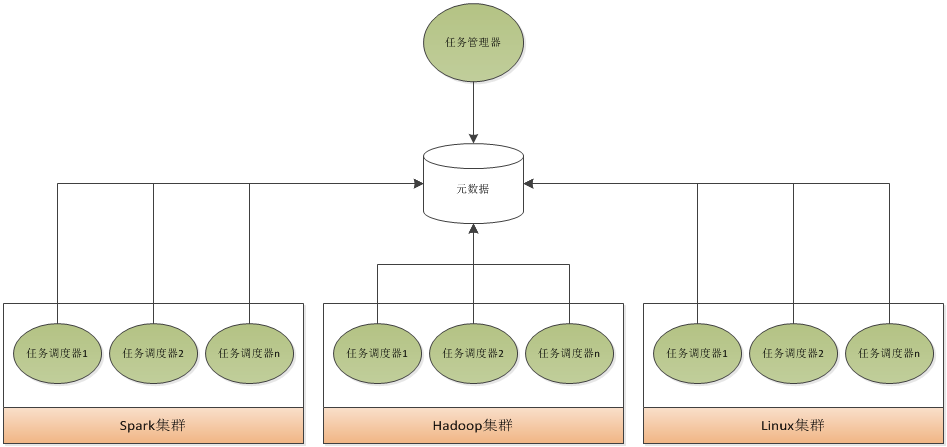


# 任务调度与监控告警

数据仓库中有很多不同类型的任务，比如数据同步任务、Spark任务、SQL任务、MDX任务、Shell任务等；这些任务除了定时调度以外，还存在着依赖关系，比如数据分析任务必须等数据收集任务完成后才能开始，数据聚合任务必须等数据分析任务完成后才能开始。

任务调度系统就是为了解决上面的问题而产生的，它作为数据仓库的中枢，负责按执行时间和依赖关系的先后调度任务。

## 调度系统架构图



## 任务管理器

任务管理器周期性的从元数据库中读取任务配置信息，根据任务的执行周期和时间生成任务实例写入任务池，检查任务池中任务的依赖关系，当依赖的任务实例都执行成功后，更新任务实例为可执行状态，并清理任务池中的历史任务实例。

任务管理器目前是单节点，但有定时任务监控任务管理器进程，一旦检测不到进程，监控程序就会重新启动任务管理器。

**任务周期**

|  |  |  |
| --- | --- | --- |
| **任务周期** | **周期值** | **说明** |
| 天 |  | 每天执行一次 |
| 周 | 周一到周日依次为1-7 | 每周执行一次 |
| 月 | 月初到月末依次为1-31 | 每月执行一次 |
| 小时 |  | 每小时执行一次 |
| 时间间隔 | 数值，单位为秒 | 每隔几秒钟执行一次 |
| 即时 |  | 立即执行 |

**任务依赖**

任务依赖关系为有向无环图：

依赖类型分为最后一个周期依赖和全周期依赖，比如任务B依赖任务A

**最后一个周期依赖**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **任务B**  **任务A** | **天** | **周** | **月** | **小时** |
| **天** | 当天 | 当天 | 当天 |  |
| **周** |  | 当周 |  |  |
| **月** |  |  | 当月 |  |
| **小时** | 当天0点 |  |  | 当前小时 |

**全周期依赖**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **任务B**  **任务A** | **天** | **周** | **月** | **小时** |
| **天** |  | 上周二 -> 当天 | 上月2号 -> 当天 |  |
| **周** |  |  |  |  |
| **月** |  |  |  |  |
| **小时** | 前一天1点 -> 当天0点 |  |  |  |

注：目前只有任务周期为“天”、“周”、“月”、“小时”的任务才支持任务依赖。

## 任务调度器

任务调度器周期性的从元数据库中读取一定数量的可执行的任务实例，并启动一个任务运行器来执行任务；任务运行器负责监控并更新任务的执行状态，并根据配置信息发送邮件、短信告警。任务失败后，由任务调度器根据任务配置的最大重试次数重启任务。当任务运行超时后，任务调度器会杀死该任务。

任务调度器调度方式为分散式调度，调度器部署在多个节点上，一个任务由集群内多个调度器竞争调度。

# 参考文献

1. <http://hadoop.apache.org/docs/r2.7.4/>
2. <http://spark.apache.org/docs/2.1.0/sql-programming-guide.html>
3. <http://kafka.apache.org/documentation/>
4. <http://flume.apache.org/FlumeUserGuide.html>
5. <http://hbase.apache.org/book.html>
6. <http://zookeeper.apache.org/doc/r3.4.10/>
7. <https://community.meteorite.bi/docs/>
8. <https://mondrian.pentaho.com/documentation/>
9. <https://msdn.microsoft.com/zh-cn/library/ms145970(v=sql.120).aspx>