第 34 卷　第 7 期　　计　算　机　工　程　　2008 年 4 月
Vol.34　No.7　　Computer Engineering　　April 2008
·网络与通信·　　文章编号：1000—3428(2008)07—0107—03　　文献标识码：A　　中图分类号：TP311.13

# 基于分布式多级目录的 NetFlow 流数据检索

贾冠昕，杨　波，彭立志
（济南大学信息科学与工程学院网络中心，济南 250022）

**摘　要：**对于网络流量工程而言，需要解决如何存储并高效地检索大量的网络流量数据。该文提出利用分布式多级目录技术实现 NetFlow 网络流数据信息存储与检索的一个完整方案，讨论 NetFlow 流数据的存储方式以及对其进行检索的优化方案，给出该系统的工作流程和框架、对流数据查询检索的部分数据结构及其工作方式。

**关键词：**NetFlow 技术；流数据；分布式；多级目录

# Distributed Multi-level Directory Based NetFlow Flow Data Retrieval

JIA Guan-xin, YANG Bo, PENG Li-zhi
(Network Information Center, School of Information Science and Engineering, Jinan University, Jinan 250022)

**【Abstract】** Because of the great quantity of network traffic data, the first problem for network traffic engineering is how to store traffic data and retrieve these data efficiently. This paper presents a design of how to store and retrieve NetFlow data in distributed multi-level directory. An optimization blueprint is discussed whose objective is to store and retrieve NetFlow data efficiently. This paper also brings out the frame of this system and part of data structures which are used to retrieve data. How to work with these data structures is also presented.

**【Key words】** NetFlow; flow data; distributed; muti-level directory

网络流量的检测分析是网络管理和系统管理的一个重要组成部分，网络流量数据为网络的运行和维护提供了重要信息。这些数据对网络的资源分布、容量规划、服务质量分析、错误监测与隔离、安全管理都十分重要。作为网络流量检测的主流技术之一，NetFlow不需要其他硬件流量设备的支持，开启和关闭非常方便[1]。目前NetFlow流数据已被用于网络入侵监测[2]、按流量计费[3]、病毒检测[4]等方面。

## 1　NetFlow 流数据的定制流程

整个 NetFlow 流数据应用的架构如图 1 所示。

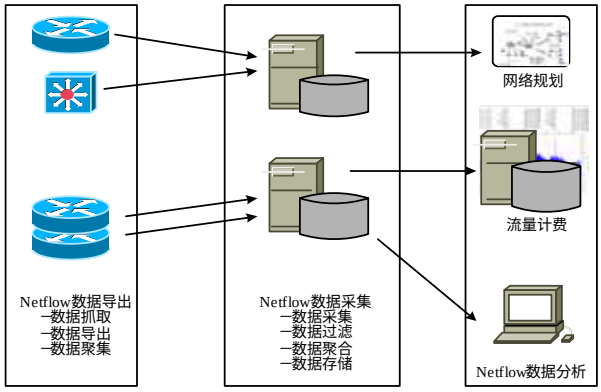


**图 1　NetFlow 流数据应用架构**

NetFlow 是按照网络流进行记录的，一个网络流是从给定的源到目的端单向的一系列数据包，它使用源和目的端点的 IP 地址和传输层端口号、协议类型、服务类型以及输入接口等来标记网络流。NetFlow 的数据输出要求先在路由器和交换机上定制 NetFlow 流输出，并选择输出流的版本、个数、缓冲区的大小等，配置相应 NetFlow 流数据采集程序的 IP 地址、端口等信息，此时路由器或交换机就可以把网络流数据经过初步的聚集后以 UDP 的方式向相应的 IP 及端口发送流信息。一般来说，NetFlow 采集程序都选用 Unix 服务器来收集数据，采集程序将收集到的流记录存放在本地磁盘中。与此同时，采集程序通过本地网关以 SOCKET 方式发送信息到其他网管分析软件，如 Cisco 公司的 Netflow FlowAnalyzer 流量分析软件。也可以直接读取存放在采集程序所在服务器中的数据文件，对其进行分析处理。后一种方法需要用户自己编写相应的流数据服务程序，以便对网络流数据的检索请求作出响应。

## 2　NetFlow 流数据的接收存储

NetFlow 流数据在路由器经过初步的聚集后就以 UDP 包的形式发送至接收服务器的相应端口，在数据接收服务器还需要再进一步的聚集、过滤后存储到本地磁盘中。文献[5]对流数据的聚集、排序等有较详细的论述。

### 2.1　流数据的存储方式

对于具备了一定规模的网络来讲，NetFlow 流数据的数据量是非常大的，如果按照常规的数据库存储方式保存流数据，数据库的插入效率、文件大小都会成为网络流量监测分析的瓶颈。最简单的解决方式就是采用二进制文件的方式保存流数据，即把每条流数据中有价值的字段以二进制的形式、固定长度按到达顺序写入流数据文件中。

### 2.2　按时间分割流数据文件

NetFlow 流数据的数据量非常大，如果把所有的数据保存在一个文件中是不可能的，而且会导致数据检索效率极度低下。仅以济南大学校园网出口路由器返回的 NetFlow 流数

**作者简介：**贾冠昕(1975－)，男，工程师、硕士研究生，主研方向：计算机网络；杨　波，教授、博士；彭立志，硕士

**收稿日期：**2007-04-13　**E-mail：**jiaguanxin@126.com

据为例，如果每一个流的数据仅保存源 IP、目的 IP、源端口、目的端口、协议类型、字节数、发生时间等字段按照二进制方式写入到文件中去，则每条流记录占用 24 B，每天向校园网内发送的网络数据流量即可达到 500 MB，如果再保存流出流量，那么每天的数据量就可以达到 1 GB 左右，这样惊人的数据量只能把数据分布到多个文件中保存。

由于对流数据的检索通常情况下都会限定在某段时间内，因此最为方便可行的方法就是按照时间段分割流数据文件。例如可以考虑将每天的数据单独保存在一个文件中，这样在 9 月 24 日发生的流数据将保存在 flow0924 这个文件中。这样如果需要检索某段时间内的流量数据，只需要检索这段时间内的文件即可。

### 2.3 多级目录存储检索流数据

对于 NetFlow 流数据这样巨大的数据量来讲，简单地按时间段分文件存储仍然无法满足检索要求。仍以济南大学校园网流量为例，即使只关心出口路由器的流量数据，一天的流入流出数据量就达到 1 GB，也就是说大约有四千多万条流记录，如果还需要关心校园网内部的数据流量的话，数据量更是非常惊人。鉴于绝大多数的流量检索都是具体到 IP 或 IP 段的，而按时间段分文件存储方式是所有这段时间内的流记录按照时间顺序保存在一个文件中的，也就是说如果需要检索某一个 IP 在一天中的流量数据，仍然要访问当日流数据文件的全部记录，检索的命中率显然非常低下。

为了解决这个问题，可以在按时间段分割文件的基础上再进一步分割文件。在一个园区网络管理中，只关心涉及本园区内部 IP 的数据流量，即关心的每个流数据的源、目的 IP 至少有一个是本地 IP。这时可以按照流量记录中的本地 IP 继续分割文件，把具备某个相同特征的本地 IP 所涉及的流数据按时间段单独存放在一个文件中。在校园网中由于申请的网络地址通常是连续的，IP 地址的前 2 个字节相同的概率非常大，因此可以按照用户 IP 地址的后 2 个字节确定所要存放的文件名。这样保存流数据会产生大量的文件，由于在同一个目录下如果文件过多会导致文件 I/O 操作效率严重下降，因此需要把按 IP 地址分割后的文件放置到特定的目录下。按照 IP 地址的最后 2 位确定该用户的流数据文件的存放目录：IP 地址的第 3 位作为第 1 级目录；第 4 位作为第 2 级目录。这时，在每个目录下的文件名无需再反映 IP 地址特征，只需反映时间特征即可。例如，9 月 24 日对 192.168.23.45(IP 地址为假设地址)作为目的地址的流数据保存在“ 流数据根目录/23/45/flow0924 ”这个文件中。实验结果证明在指定 IP 地址或 IP 地址段的情况下，检索效率的提高是指数级的。

## 3 分布式处理 NetFlow 流数据的存储检索

如前所述，NetFlow 流数据的数据量是如此的巨大，即使采取了前面所提到的方法仍然面临着很多问题：首先，每天的数据量太大，如果保存在一台数据服务器上对于硬盘的要求非常高。以济南大学保存一年的流数据为例，很可能达到 400 GB 左右，如果再保存校园网内部的流量数据的话，数据量会更加惊人，而且随着校园网的发展这个数据还会继续增加。把这样海量的数据保存在一台数据服务器上是非常困难的，检索效率也难以保证。其次，当对 NetFlow 流数据的检索请求增多时，数据服务器的效率将成为瓶颈。

### 3.1 分布式存储流数据

这些问题通过分布式服务方式可以得到解决，即多台数据服务器协同工作，把流数据的存储与检索分流到各台流数据服务器上来解决 NetFlow 流数据服务器端的瓶颈问题。这时整个 NetFlow 流数据处理系统可以分为 4 个部分：

(1)路由器(router)；

(2)流转发服务器(flow relay server)；

(3)数据服务器(data server)；

(4)请求转发服务器(request relay server)。路由器所产生的所有 NetFlow 流数据首先发往流转发服务器，流转发服务器视各数据服务器的负载情况决定数据转发往哪台数据服务器保存。

### 3.2 数据服务器负载平衡问题

流转发服务器在决定流数据的转发目的地时，主要考虑各数据服务器的负载情况。为了获取数据服务器的负载情况，每个数据服务器设立“ 负载报告进程 ”专门负责报告本地资源使用情况。该进程主要用于监听特定的端口并向流转发服务器反馈硬盘空闲大小(*diskFree*)、正在处理用户查询请求剩余检索量(*queryRemained*)以及已接收到正在排队等待写入的流数据链表长度(*waitingLen*)等这些信息。

流转发服务器通过维护一个数据服务器列表取得各个数据服务器的信息，定期轮询各数据服务器获得前面所述的负载信息(如：10 次/s)，通过负载平衡函数计算出各数据服务器的负载系数。数据服务器 $i$ 负载系数如下：

$$Load_i = \frac{\mathrm{lb}(diskFree_i)}{query\,\mathrm{Re}mained_i + waitingLen_i + 1}$$

根据各数据服务器负载系数计算出流数据转发往第 $i$ 台的概率为

$$p_i = \frac{load_i}{\sum_{j=1}^{n} load_j}$$

通过上述方式可以平衡各台数据服务器的负载量，获得更佳的效率。

### 3.3 处理用户检索请求

用户计算机在提交数据检索请求之前监听结果接收端口，该端口用于接收数据返回的检索结果。用户提起的检索请求首先发往请求转发服务器，请求转发服务器负责把用户请求转发到各台数据服务器，由各数据服务器生成相应的检索结果并直接发送给用户的结果接收端口。在这里用户端只需要知道请求转发服务器的存在即可，无需知道其他设备的信息。由于流记录是按照概率随机发往各台数据服务器的，因此每次用户发起检索请求，请求转发服务器都会把该请求以及用户的 IP、结果接收端口等信息转发给所有的数据服务器。每台服务器分别返回检索结果，做到了数据的分布式并行处理。通过这种方式，整个 NetFlow 流数据的处理分布到了多台数据服务器中，大大减轻了数据服务器的压力。更为重要的是整个系统的可扩展性大大增强了，随着网络规模的增加，可以通过增加数据服务器数量提高流数据的处理能力。

此时，整个流数据服务器端的运行流程如图 2 所示，整个系统的架构框架如图 3 所示。图 3 中只表明了逻辑上的层次，并不是每层的设备一定按照图中所示的设备配置。对于压力不大的部分设备完全可以合并，而对于压力过大的设备可通过增加设备量来缓解压力。例如，对于一般情况下的索引服务器，如果压力不大，则可以与某台同样压力不大的数据服务器合并；同样的，如果转发服务器压力大，可以增加一台转发服务器，使各个路由器产生的 NetFlow 流数据分流到 2 台转发服务器上去。这种构架的好处是各层之间是分离的，互不干涉。各层之间只需要按照约定的规范接受请求和

发回结果，而不需要关心其他层的具体实现，更加灵活。事实上，这里的用户是相对于流记录文件的操作而言的，用户层可能是另一个集成了流量分析功能的服务程序，对于最终用户这里提到的客户端程序仍然是在远端服务器上运行的。

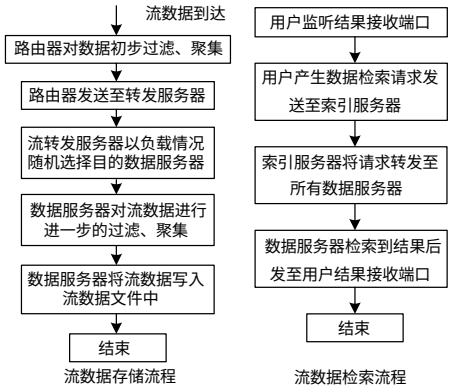


**图 2　服务器端的运行流程**

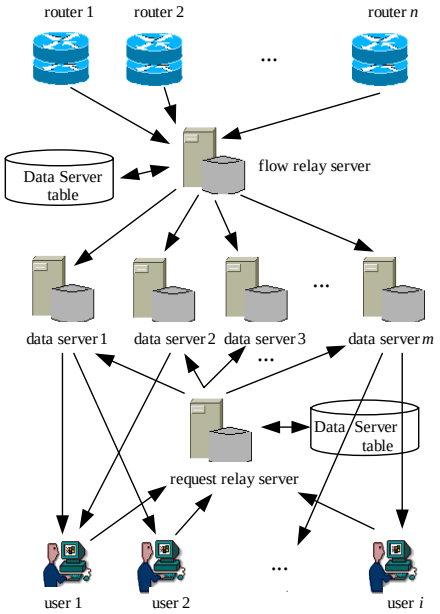


**图 3　系统框架图**

## 4　其他相关问题

在前面提到的方案中还存在某些问题需要解决：各个层之间的通信通过何种方式进行，如何传输数据。

除路由器与转发服务器之间的通信采用 NetFlow 的规范发送数据外，其他各层之间都可以采用用户自定义的应用层协议。下面给出已经应用于实际项目的自定义协议的一部分予以说明(出于效率的考虑，各层之间的通信全部以二进制传输)：

用户端向索引服务器发送请求的格式：

```
struct query_packet{
        int protno:8888;          //所有通信的起始数据包都是
//以固定的整数 8888 作为标志
        int type:1;               //该数据包的类型，这里 1 代表
//是检索请求数据包
        time_t time_start;        //要检索的流数据的起始时间
        time_t time_end;          //要检索的流数据的结束时间
        uint32_t  srcIPStart;     //要检索的流数据的源 IP 的 IP
//段起始 IP
        uint32_t srcIPEnd;        //要检索的流数据的源 IP 的 IP
//段终止 IP
        uint32_t dstIPStart;      //要检索的流数据的目的 IP 的
//IP 段起始 IP
        uint32_t dstIPEnd;        //要检索的流数据的目的 IP 的
//IP 段终止 IP
        unsigned short srcPort;   //要检索的流数据的起始端口
        unsigned short dstPort;   //要检索的流数据的终止端口
        unsigned char prot;       //要检索的流数据的通信协议，
//如 TCP：6，UDP：17
        unsigned char recordType; //要检索的流数据的类型：0:
//国内流量，1:国际流量
        unsigned short resultPort; //本次检索请求所监听的结果
//接收端口
    };
```

数据服务器向用户发送结果的格式：

```
struct result{
        int protno:8888;          //所有通信的起始数据包都
//是以固定的整数 8888 作为标志
        int type:2;               //该数据包的类型，这里 2 代
//表是检索结果数据包
        int flowNum;              //本次传输的后续流数据数量
};
```

发送完这个包后立刻向用户发送 flowNum 条流数据，结构如下：

```
struct flow{
        uint32_t srcIP;           //流记录的源 IP
        uint32_t dstIP;           //流记录的目的 IP
        unsigned short srcPort;   //流记录的源端口
        unsigned short dstPort;   //流记录的目的端口
        unsigned char prot;       //流记录的协议类型
        uint32_t pkts;            //流记录包含的数据包数量
        uint32_t bytes;           //流记录的字节数
};
```

使用上述自定义的应用层协议可实现用户发起流数据检索请求以及数据服务器返回检索结果的功能。至于其他各层之间的通信可采用类似的方法实现，无需再列出具体的实现。

## 5　结束语

NetFlow 流数据的海量特征，使得所有针对流数据的操作都要考虑效率因素。通过分布式多级目录实现的 Netflow 流数据存储与检索系统，基本可以满足对 NetFlow 流数据的普通检索要求。如何利用 Cache 技术增加查询效率以及是否可以建立高效的索引机制都是需要进一步研究的问题。

## 参考文献


[1] 谢喜秋, 梁　洁. NetFlow 服务对网络性能的影响[J]. 广东通信技术, 2002, 22(3): 39-44.

[2] 黄　艳, 李家滨. 基于 NetFlow 的网络入侵监测系统[J]. 计算机应用与软件, 2006, 23(6): 85-86.

[3] 吴　斌, 丘劲松, 金连甫, 等. 基于 NetFlow 的流量计费系统的设计与实现[J]. 计算机工程, 2004, 30(7): 189-191.

[4] 黄家林, 张　超. 基于 Netflow 数据流的蠕虫探测算法[J]. 网络安全技术与应用, 2005, 10(1): 49-50.

[5] 王　华. 软件实现 Netflow 流量处理的关键技术和算法[J]. 计算机工程, 2004, 30(增刊): 232-234.