第 38 卷 第 3 期
Vol.38 No.3

计 算 机 工 程
Computer Engineering

2012 年 2 月
February 2012

• 软件技术与数据库 •　　文章编号：1000—3428(2012)03—0043—03　　文献标识码：A　　中图分类号：TP311.5

基于文本策略和 SMCS 的海量日志分析方法

张俊峰[1]，冯巧娟[1]，张晓丽[1,2]

(1. 河南城建学院计算机科学与工程系，河南 平顶山 467036；
2. 南京航空航天大学航空科技智能材料与结构重点实验室，南京 210016)

摘 要：现有的海量日志统计分析方法速度慢，且对硬件配置的要求高。为此，提出一种基于文本策略和 SMCS 的海量日志分析方法。根据文件的软件设计策略，采用日志文件索引方法，将日志文件与日志时间关联，以加快日志提取。SMCS 算法采用哈希表、文件归并、堆操作方法对海量日志进行统计分析和内存损耗控制。通过对真实软件进行对比实验，结果表明，该方法的分析速度比传统方法提高 4 倍。

关键词：Syslog 日志；日志分析；SMCS 算法；海量日志；文本策略；控制内存

Mass Log Analysis Method Based on File Strategy and SMCS

ZHANG Jun-feng[1], FENG Qiao-juan[1], ZHANG Xiao-li[1,2]

(1. Department of Computer Science and Engineering, Henan University of Urban Construction, Pingdingshan 467036, China;
2. Aeronautical Science Key Laboratory for Smart Material and Structures,
Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China)

【Abstract】Now the statistical analysis for the massive logs not only is time-consuming, but also requires higher hard configuration. For these questions, a method based on the file software strategy and SMCS algorithm is proposed in this paper. By the software strategy, the log file indexing and the association between the log file and log time are adopted in this method, so as to quicken the log extraction. In order to analyze the massive logs statistically and control the memory loss, hash tables, file merging and heap operation are used in the SMCS algorithm. By experiment of the real software, results indicate that the proposed method is faster than that of traditional statistical analysis by four times.

【Key words】Syslog log; log analysis; SMCS algorithm; mass log; file strategy; control memory

DOI: 10.3969/j.issn.1000-3428.2012.03.015

1 概述

路由器是各种信息传输的枢纽，承担着局域网之间及局域网与广域网之间连接的重任。路由器产生的 Syslog 日志会传给日志处理终端，Syslog 日志格式符合 Syslog 协议[1]。日志详细记录各次连接信息。通过日志分析软件分析日志，可以检查网络发生攻击的原因、定位攻击者的各种信息等。因此，这些日志对于网络安全具有重要意义。为了应对不同的网络威胁，越来越多的安全防御平台都在不断加强内容安全防御功能[2]，日志分析软件也越来越受到重视。

在大型网络中(如高校)，路由器每天产生的日志高达 100 GB(纯文本)。海量日志对日志分析软件的设计策略提出更高的要求。如下为某品牌路由器产生的一条 Syslog 日志，日志详细记录一次连接信息：

05-19-2009 11:29:26192.168.0.1 local0.info [2009-05-19 11:29:01] EFW: CONN:
prio=1 id=00600004 rev=1 event=conn_open_natsat
rule=lan_to_all conn=open connipproto=UDP
connrecvif=int connsrcip=192.168.0.149 connsrcport=1067
conndestif=ext conndestip=61.139.2.69 conndestport=53
connnewsrcip=222.212.78.105 connnewsrcport=4152
connnewdestip=61.139.2.69 connnewdestport=53

从上面日志得到的信息有：<192.168.0.149(源 IP)、61.139.2.69(目的 IP)>发生了一次；<192.168.0.149(源 IP)、1067(源端口)、61.139.2.69、53(目的 IP)、53(目的端口)>发生了一次；日志产生的时间为：2009 年 5 月 19 号，11 点 29 分 1 秒等信息(注：许多信息没有在此条日志反映出来)。一个厂商的日志种类很多，每种日志反映的信息不一样。

日志分析软件需要有日志统计分析和条件检索功能[3]。如：查找某个 IP 所有连接信息；统计发生次数最多的<源 IP、目的 IP、目的端口>(前 400 位)等。可以作为检索条件有：源 IP，目的 IP，源端口，目的端口，事件严重级别等。

目前，国外有名的日志分析软件为 EIQ 公司的 Syslog Insight 软件。国内同类软件无法在大型网络环境使用。为了弥补国内软件不足，本文提出了一种基于文本策略和 SMCS 的海量日志分析方法。

2 软件设计策略

2.1 基于数据库的设计策略

统计分析 Syslog 日志最有效的思想是借助于数据库。思想为，日志解析器解析路由器发送过来的每一条日志，并提取各个信息，如日志产生的时间、源 IP、目的 IP、目的端口、Nat 地址等。将提取的信息批量存入数据库相应表中。基于数据库的策略软件框架如图 1 所示。数据库设计成为影响整个软件性能的重要因素。将数据库设计成一个主表，主要包括字段有(已略去一些字段)：日志厂商类型标识，日志 id(与厂商有关)，日志产生时间，产生日志的路由器 IP 地址，日志的类型，日志源 IP，源端口，目的 IP，目的端口，NAT 源

基金项目：国家自然科学基金资助项目(60907038)；河南省科技攻关计划基金资助重点项目(102102210020)

作者简介：张俊峰(1967－)，男，副教授、硕士，主研方向：网络安全，分布式处理；冯巧娟，讲师、硕士；张晓丽，讲师、博士

收稿日期：2011-08-17　**E-mail：**pzjf2010@163.com

IP，NAT 源端口，NAT 目的 IP，NAT 目的端口等。

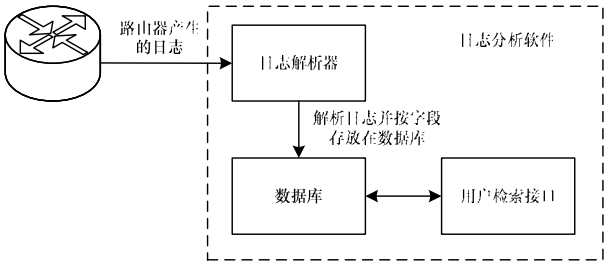

图 1 基于数据库的策略软件框架

将数据库设计成一个表，有如下原因：

(1)一个表已经符合数据库设计第三范式[4]，主键为<日志厂商类型标识、日志 id(与厂商有关)>。

(2)设计成一个表，极大地加快数据库检索及统计速度；对提取之后的日志，能很方便地进行批量存取。

基于数据库的策略，最大缺点是统计性能受限于数据库；性能较好的数据库为 Oracle 数据库，但是成本太高，并不适用于软件部署[5]。

2.2 基于文件的设计策略

基于文件的设计策略思想为：软件分析器将日志解释之后存储为二进制文件，并建立文本索引文件将二进制文件名和日志产生时间范围对应起来，以加快对日志的统计分析。基于文件的策略软件框架如图 2 所示。

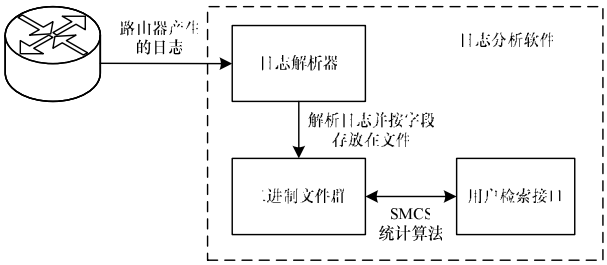

图 2 基于文件的策略软件框架

将日志存在多个二进制文件：加快对指定时间内的日志提取，并加快统计速度。一个二进制文件存放的日志量达到一定数量时(1 000 万条)，则将新到来的日志存储在一个新的二进制文件。二进制文件与索引文件的索引情况见图 3。

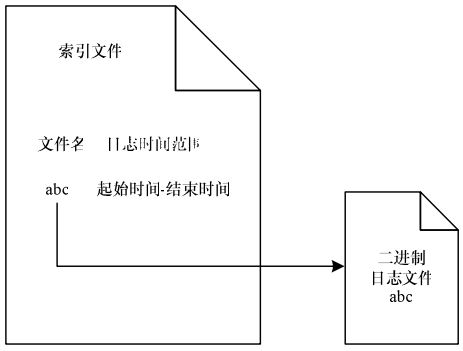

图 3 索引文件与日志文件关系

索引文件第 1 列为二进制日志文件名；第 2 列为日志文件中所有日志对应的时间范围。

在分析日志时，通过索引文件提取用户指定时间范围的日志文件，批量地读取日志，并交由 Syslog 内存控制统计(Syslog Memory Control Statistical, SMCS)算法进行统计分析。

3 SMCS 算法

以统计发生次数最多的<源 IP、目的 IP、目的端口>(前 400 位)为例介绍 SMCS 算法。

3.1 变量定义

设 Key 为<源 IP、目的 IP、目的端口>的组合结构体。

```
struct Key{
  uint32 sourceIp;//源 IP
  uint32 destinationIp;//目的 IP
  uint8   destinationPort;//目的端口};
```

Key 实例值将作为哈希表的键值。使 Key 值满足符号“<”的定义。如(key1、key2 为 Key 实例)，key1(192.168.0.2、202.202.0.3、80)<key2(192.168.0.2、202.202.0.5、80)。

Log 为日志结构体。

MaxNumber 为存储在 hashTable 的最大日志条数(根据软件运行环境动态确定)。存储临时结果文件名为 log$N$(N=1, 2,…,n)。

结果模板结构体为：

```
template <typename K,typename V>
struct ResultValue{
  K key;
  V value;};
typedef ResultValue<Key,uint64> ResultStruct;
```

ResultNumber 为 ResultSet 结果集的容量。

日志输出结果集为 ResultSet：ResultSet 中存放结构为 ResultValue <K,V>实例；需要将产生的所有 ResultValue <K,V>实例按 value 值排序，并将前 ResultNumber 个实例存入 ResultSet 中。

3.2 SMCS 算法流程

输入 从文件中读入经过过滤的日志

输出 ResultSet 结果集

```
Log   log; //日志结构体临时变量
Key   varibel;//临时变量
Map[6]<Key,uint64> hashTable;
vector[6]<string[6]> fileName;//存放临时文件名字的容器
vector<string> fileName2;//存放临时文件名字的容器
string resultFileName;//最终临时文件名
//(1)对原始日志进行处理
while(日志还有没处理完){
  得到一条日志，并赋给 log;
  将 log 中对应的值赋给 varibel;
       //统计发生次数最多的<源 IP、目的 IP、目的端口>
  if(hastTable 存在和 varibel 一样的键值){
        hastTable[varibel]映射值加 1;
        continue;
  }
  hashTable[varibel]  映射值为 1;
  if(hashTable 中的实例计数>MaxNumber){
        创建一个新的二进制文件 logN;
        将 hashTable 键值按从大到小排序，并将 hashTable 中所有的值写入文件 logN 中;
        清空 hashTable 中的所有值;
        将二进制文件 logN 文件名插入 fileName 容器中;
  }
      //发生次数最多的<源 IP、目的 IP、目的端口>  结束
     //为了不用多次读取日志文件，可以增加其他类型的统计及
//查询
  }
  if(hastTable 中的实例计数>0){
```

```
    创建一个新的二进制文件 logN;
    将 hashTable 键值按从大到小排序，并将 hashTable 中所有的值写入文件 logN 中;
    清空 hashTable 中的所有值;
    将二进制文件 logN 文件名插入 fileName 容器中;
  }
   //其他类型的统计及查询，做和上面类似处理
  //(2)对文件进行归并处理
  ResultStruct a;
  ResultStruct b;
  while(fileName 容器的大小>1){
    while(fileName 容器大于>1){
        从 fileName 容器中弹出 2 个文件名，file1,file2;
       //合并 file1,file2 把结果存在一个新的文件中 file3;
        读取 file1 文件的一行，赋给 a;
        读取 file2 文件的一行，赋给 b;
        while(file1 没有到文件尾&&file2 没有到文件尾){
            if(a.key 等于 b.key){
                ResultStruct c;
                c.key=a.key;
                c.value=a.value+b.value;
                将 c 写入文件 file3 中;
                读取 file1 的一行，赋给 a;
                读取 file2 的一行，赋给 b;
                continue;
            }
            if(a.key 大于 b.key){
                将 a 写入文件 file3 中;
                读取 file1 的一行，赋给 a;
                continue;
            }else{
                将 b 写入文件 file3 中;
                读取 file2 的一行，赋给 b;
            }
        }
        处理没有到文件尾的文件，将数据写入 file3 中;
        将 file3 文件名存入 fileName2 容器中:
        删除文件 file1,file2;
    }
    将 fileName2 容器赋给 fileName 容器;
    清空 fileName2 容器
  }
  //其他类型的统计及查询，做和上面类似处理
  //(3)求出结果
  if( fileName 容器的大小==1){
    fileName 容器的值赋给 resultName;
  }else(fileName 容器的大小==0)
    return;
  构造一个小堆 ResultSet
  //从 resultName 文件中一条一条读取 ResultStruct 实例
  while(没有到达文件结尾){
    读取 Key2 实例，并赋给 tmp;
    if(ResultSet 的大小<ResultNumber){
        将 tmp 插入小堆中;
        continue;
    }
    if(tmp>ResultSet 的堆头元素){
        将 tmp 替换堆头元素，重新调整小堆;
    }
  }
  将 ResultSet 进行堆排序; 并输出;
  //其他类型的统计及查询，做和上面类似处理
```

算法分析：假设日志条数为 n，整个算法分为 3 个部分。

(1)控制 hashTable 的大小，日志软件进行统计分析时，会存在多个 hashTable；通过对 hashTable 容量进行控制，达到对内存进行控制目的，因为 hashTable 键值需要进行排序输出到文件，所以时间复杂度为 $O(n\text{lb}n)$。

(2)对各个分开的文件进行合并，采用的是合并算法[7]，时间复杂度为 $O(n\text{lb}n)$。

(3)算出结果集，采用的是堆算法[8]，算法的时间复杂度为 $O(n\times\text{ResultNumber}\times\text{lb}(\text{ResultNumber}))$。

整个算法复杂度为 $O(n\text{lb}n)$。通过在算法中插入其他的统计模块，可以减少读取文件的次数，提升统计分析速度。

4　软件性能对比分析

对开发出的 2 款软件：基于数据库策略的软件和基于文件策略的软件(统计分析算法采用 SMCS 算法)进行测试。

2 款软件同时统计如下需求：统计流量最大的源 IP(前 400 位)；统计流量最大的目的 IP(前 400 位)；统计发生次数最多的<源 IP、目的 IP>(前 400 位)；统计发生次数最多的<源 IP、目的 IP、目的端口>(前 400 位)；统计发生次数最多的<源 IP、目的 IP、源端口、目的端口>(前 400 位)。

实验主机硬配置为：4 GB 内存，双核处理器，Windows XP 系统，数据库版本为 SQL Server2008。

实验结果如图 4 和图 5 所示。当日志量达到 13 亿多时，用数据库版本软件进行处理，机器几乎瘫痪，统计消耗时间数据无法给出。

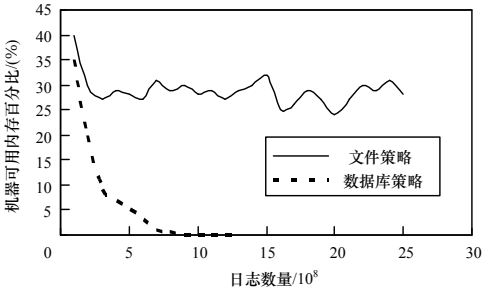

图 4　统计分析日志时机器可用内存情况

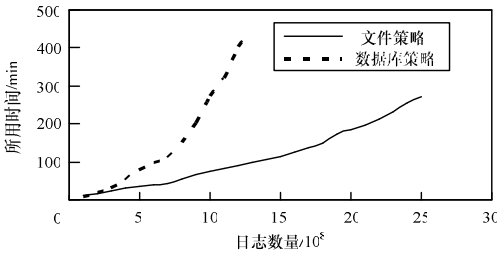

图 5　统计查询日志消耗时间

从图 4 可以得出，SCMS 算法控制了机器内存的消耗，保证了机器整体性能。从图 5 可以得出，第 2 个版本依赖于 SCMS 算法，其分析统计速度有明显改变。

5　结束语

本文给出了 Syslog 日志分析软件的 2 种不同设计策略，并对分析速度和消耗内存情况进行对比。结果证明，基于文件的设计策略可以分析海量日志，适用于大型网络。SMCS 算法对于日志分析速度和机器的稳定性起到了关键作用。

(下转第 48 页)