doi:10.11959/j.issn.1000-436x.2015300

# 基于差分压缩的大规模日志压缩系统

唐球[1,2,3]，姜磊[1,2]，戴琼[1,2]

(1. 信息内容安全技术国家工程实验室，北京 100093; 2. 中国科学院 信息工程研究所，北京 100093; 3. 中国科学院大学，北京 100049)

**摘 要**：大型信息系统的日志数据规模呈现快速增长趋势，导致线速压缩与存储大规模日志数据成为当今数据管理的一大挑战。对大量的网络系统日志进行了研究，发现日志数据存在冗余的结构模式，在内容上存在时间局部相似性。提出了基于模板的细粒度日志差分压缩架构，针对具体日志数据，可配置与其相适应的细粒度差分策略。实验结果表明，与 gzip 工具相比，所提日志压缩系统在压缩速度上提高了 2~10 倍，压缩率比 gzip 更低，可达到 10%。

**关键词**：日志；差分压缩；细粒度；模板

**中图分类号**：TP302 **文献标识码**：A

# Large-scale log compressing system based on differential compression

TANG Qiu[1,2,3], JIANG Lei[1,2], DAI Qiong[1,2]

(1. National Engineering Laboratory for Information Security Technologies, Beijing 100093, China;
2. Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China;
3. University of Chinese Academy of Sciences, Beijing 100049)

**Abstract:** The scale of log data produced by the large scale information system is growing rapidly. It leads to the big challenge of line-speed compressing and saving the large scale log data. By analysis on massive network log data, it is found that the log data has redundant pattern in terms of log structure and time local similarity in terms of log content. A differential log compression architecture based on template is proposed. Fine-grained differential compressive strategies in the architecture can be configured for a special log data. Experimental results show that, compared with gizp, the proposed log compressing architecture improves 2~10 times' compressive speed and gain a better compressing ratio approaching to 10%.

**Key words:** log; differential compression; fine grain; template

## 1 引言

随着计算机技术与网络应用的快速发展，企业或机构内部部署了多套网络设备与信息系统。这些设备或系统不间断地将各自的运行状态记录为日志数据，如 Web 服务日志、防火墙日志、入侵检测系统日志等。日志数据是服务改进、系统审计、安全分析、数据挖掘等应用的重要数据源[1~4]，因此需要长期存储信息系统的日志数据。随着大数据、移动互联网时代的到来，信息系统变得日益复杂，面向互联网提供服务的信息系统访问量急剧增长，随之而来的系统日志数据也呈现出爆炸式增长。尤其对于一些核心系统模块，需要记录其所发生的一切操作并将其传输回数据中心进行存储与分析，海量的日志数据对于日志系统的设计是一个巨大的挑战。因此，有效的压缩、传输并存储大规模日志数据是现代信息系统的迫切需求。

在数据规模较小且可定期删除历史日志数据的情况下，可以使用传统的压缩工具定期进行日志数据的压缩存储，如 Linux 日志管理工具 logrotate，

**收稿日期**：2015-11-11
**基金项目**：中科院战略性先导科技专项基金资助项目（XDA06031000）
**Foundation Item:** Special Pilot Research of the Chinese Academy of Sciences (XDA06031000)

它使用 gzip 定期对日志数据进行压缩与轮询存储。但是，使用传统的通用压缩算法降低日志数据存储空间存在效率低的问题。一方面，传统压缩算法的通用性导致了其压缩率低，因为通用的压缩算法没有充分利用日志数据的自身特性；另一方面，通用压缩算法计算量大，难以线速处理大规模的日志数据。近年来，针对大规模日志数据的压缩存储，国内外开展了大量相关工作。文献[5]利用 Web 日志数据在结构与内容方面均存在大量相似性的特性，提出了一种日志数据变换机制，将当前日志中与前一条日志相同的数据块转换为相同数据块的长度值；不相同的数据块则保留；然后对变换后的日志数据再使用通用的压缩算法做进一步的压缩；该方法的不足是对于不同类型的日志数据块采用同样的压缩算法。文献[6~8]先后尝试字段合并、不同字段差异化压缩策略等技术对 Apache 的 Web 日志进行差分压缩，与本文提出的方法的思想有一定的相似之处，但是该文献中的方法缺少可配置的灵活性，而且是专门针对于 Apache 的 Web 日志，缺少对更广泛类型日志数据的通用性设计。Kimmo 等提出了基于日志数据中的频繁重复数据模式块消冗的防火墙日志压缩算法[9]。文献[10]通过分析 DNS 日志数据的特性，针对 DNS 日志数据所涉及的时间、IP 地址、域名和类型 4 类数据分别进行不同的压缩方法。文献[11]提出了基于通用压缩算法压缩日志数据的预处理算法，通过预处理算法中的分区函数将日志数据划分至多个同质日志桶（homogeneous buckets），然后对不同的同质日志桶内的日志数据使用传统的压缩算法 bzip2 或 gzip 进行压缩。由于同质日志桶内容日志结构高度相似，因此比直接使用 bzip2 或 gzip 压缩压缩更高。文献[12]使用硬件 FPGA 实现 LZ4 算法对外汇交易系统中的事务日志进行压缩，在保证 LZ4 的压缩率的情况下，获得了专用硬件带来的压缩速度优势。已有日志压缩工作降低了日志存储空间，但是存在以下几点不足：1）通用性较差，依赖于具体的某一类日志；2）不具可扩展性，不支持压缩策略的可配置；3）粗粒度压缩策略，针对不同数据类型采用相同的压缩策略；4）不支持流式差分，难以满足在线实时日志压缩需求。

本文针对以上不足，提出了一种支持大规模日志数据的流式差分压缩架构。该架构首先充分利用日志数据的先验知识，将一类日志数据中固定不变的数据块提取为模板，日志压缩时，删除模板中定义的数据块，只存储一个模板指针。在解压时，根据模板内容即可复原日志数据；然后，针对日志数据中不同类型的数据块定制其差分压缩策略。本文提出大规模日志压缩系统不依赖于特定的日志类型，对于不同类型的日志数据可以配置不同的细粒度差分压缩策略，该系统采用流式的差分压缩架构，从而使本文提出的日志压缩系统具有明显的压缩率与压缩速度优势。

## 2 日志数据分析

日志数据格式可以是系统设计者自定义，或使用被广泛使用的 syslog 日志格式[13]，后者由 RFC3164 规范定义。它规定一条日志消息由“优先级”、“头部”与“消息体”3 部分构成，其中优先级是一个数字，它代表了生成日志的程序模块（facility）与严重性（severity）；头部包含时间与主机名；消息体是具体的日志内容。由于 syslog 日志规范只是一个建议，规范宽泛，其消息体包含了大量的日志属性，但这些内容的组织格式并没有确定的定义，由系统设计者自行定义。所以，很多日志系统是将其作为一种支持的格式，更多的是对它进行扩展。如防火墙安全领域更多的是使用 NetIQ 公司提出的 WELF 日志格式。从真实的应用系统中采集了 21 GB 的真实日志数据，这个数据集由真实系统中的防火墙（天融信、思科）日志、交换机（华为）日志，入侵检测系统（IDS）日志、VPN 日志与病毒检测系统日志构成。通过对该数据集分析，发现目前的日志系统存在以下 3 个显著特征。

1) 日志格式规整相似。目前成熟产品的日志数据格式类似于 syslog RFC3164 规范的格式，或者更加规范的日志格式，如 WELF。前者如思科防火墙的日志格式（如表 1 所示）；后者的一条日志消息通常是由多个“<字段名，字段值>”的键值对构成，不同日志系统仅在键值连接符与键值对之间的分割符上有细微的差别，如表 1 中的最后 3 条日志数据，天融信的防火墙日志的键值对分隔符为空格，键值连接符为等号，而绿盟的 IDS 日志的分隔符为分号，键值连接符为分号。

2) 日志数据存在冗余结构模式。从表 1 中可知，同一类日志数据无论以哪种格式表示，在消息结构上都存在大量的相同模式，如思科防火墙的 106017 类型日志（表 1 中的第 1、第 2 行）均含有“ASA*Deny IP due to Land Attack from*to*”的模

式；天融信防火墙的访问控制日志（表 1 的第 3、第 4 行）均含有 “id=*time=*fw=*pri=*…”模式。其中模式中的星号表示可变内容，非星号字符为恒定不变内容（即冗余的结构模式）。

3) 日志数据中的属性值类型是固定且具有很强的时间局部性相似性。即对于同一类日志数据，即使在数据组织格式上有差异，但所承载的核心信息是相同的。如不同供应商的防火墙设备生成的访问控制日志数据均包含某一次穿越防火墙连接的五元组、时间、MAC 地址、生成日志的设备名、防火墙策略 ID 号等信息；在一个具体的日志系统中这些属性值的类型是确定的，且在短时间内，这些属性值是局部相似的，如时间属性值只是秒数的差异、少数 IP 地址在五元组中频繁重复出现等。

**表 1　常见网络设备日志数据实例（隐私信息已做替换）**

| 日志 | 数据 |
|---|---|
| 思科防火墙日志 | Apr 29 2015 23:49:55 ASA5585: %ASA-2-106017: Deny IP due to Land Attack from 10.0.1.5 to10.0.1.6<br>Apr 29 2015 23:00:09 ASA5585 : %ASA-2-106017: Deny IP due to Land Attack from 10.0.1.7 to10.0.1.8 |
| 天融信防火墙日志 | id=tos time="2015-04-30 08:01:49" fw=TopsecOS pri=6 type=conn recorder=session src=10.0.1.5 dst=10.0.1.6 proto=tcp sport=49726 dport=10050 inpkt=9<br>outpkt=10 ⋯<br>id=tos time="2015-04-30 15:01:49" fw=TopsecOS pri=6 type=conn recorder=session src=10.0.1.5 dst=10.0.1.6<br>proto=tcp sport=2450 dport=88 inpkt=5 outpkt=4 ⋯ |
| 绿盟 IDS 日志 | time:2015-04-30 12:11:02;danger_degree:1;breaking_sighn:0;<br>event:[30060]DNS 服务获取服务器版本号请求操作;<br>src_addr: 10.0.1.5;src_port:58729;dst_addr: 10.0.1.6 ⋯ |

## 3　日志数据差分压缩

### 3.1　基于模板的日志去重

基于第 2 节关于 21 GB 日志数据的分析结论，本节提出一种基于模板的日志消冗机制，消除日志数据中存在的冗余结构模式，形式化定义如下。

**定义 1**（字段，*field*）。字段是由字段名（*fld*）与字段值（*val*）构成的键值对（*fld*λ*val*），其中“λ”为字段名与字段值之间的连接符。一个字段表示一个具体的日志属性，如表 1 中的天融信防火墙访问控制日志中的源 IP 地址字段为“src= x.x.x.x”，其中 src 为字段名（源 IP），“x.x.x.x”为字段值（IP 地址），字段名与字段值连接符为等号。

**定义 2**（日志消息, *log_msg*）。一条日志消息表示一个事件，它是由有限个字段构成，字段之间有先后顺序、由字段分隔符（θ）相连接；即 $log\_msg=field_1\theta field_2\theta\cdots field_N$，其中，$field_i=fld_i\lambda val_i$。

**定义 3**　（日志，*log*）。日志是由多条日志消息构成的集合，$log=\{log\_msg_i \mid i\in 1,2,3,\cdots,M;\ log\_msg_i=field_1^i\theta field_2^i\theta\cdots field_N^i;\ field_j^i=fld_j^i\lambda val_j^i,\ j\in 1,2,\cdots,N\}$。

由前一节的分析可知，已有的日志格式与本文形式化定义的日志格式很相似，因此只需对待压缩的日志数据进行简单预处理，将日志数据转换为规整的日志格式(*log*)。日志数据所有的字符都是可打印字符，不含控制类字符，因此可以选择控制类字符作为 λ 与 θ。对于同一类日志数据，可以通过正则表达式匹配加文本处理脚本完成日志预处理。

同一设备或系统的日志存在冗余的结构模式：日志消息的字段组成结构是固定的，且各个字段名是相同的，即 $log\_msg_1$ 与 $log\_msg_2$ 都包含相同的内容：$fld_1\lambda,\theta fld_2\lambda\cdots\theta fld_N$；不同的部分是：$val_1, val_2,\cdots,val_N$。基于此特征，本文提出的日志压缩系统将日志中冗余的结构模式以字段为单位提取为模板，然后将模板定义的冗余模式从日志消息中删除，并在日志消息中存储模板指针。具体的模板形式化描述如下。

**定义 4**　（模板，*template*）。一个模板由字段名与字段值连接符、字段分割符、各个字段名以及该模板的 ID 号（*tid*）构成，即 $template=\{tid, \lambda, \theta, fld_1, fld_2,\cdots, fld_N\}$。

因此，基于模板的日志去重机制主要步骤如下。

**Step1**　日志预处理。

**Step2**　对每一条日志消息 $log\_msg_i$ 进行如下操作。

1) 模板库中查找与 $log\_msg_i$ 相匹配的模板（$template_k$）；

2) 将 $template_k$ 定义的字段名从 $log\_msg_i$ 中删除，并将 $template_k$ 的模板 ID 写入日志消息中，最后得到基于模板变换后 $log\_msg_i$ 为“$tid_k\theta val_1^i\theta val_2^i\theta\cdots\theta val_N^i$”。

对于压缩后的日志数据，根据模板指针，通过执行基于模板的逆去重操作即可恢复原始的日志数据。

### 3.2　日志数据细粒度差分压缩

基于模板的日志数据重后降低了日志的存储空间，但仍存在大量的信息冗余，尤其是短时间内的同类日志消息字段之间存在局部相似性。为了进

一步压缩日志数据空间，本文设计的日志差分架构对基于模板去重后的日志数据做进一步的细粒度的差分压缩。首先定义几类通用的差分压缩策略（*diff_strgy*）；然后，根据不同的字段值特性，选择各个字段值最适合的差分策略对基于模板去重后的日志数据进行字段级别的差分压缩。为了支持日志消息的线速压缩，日志差分压缩只选择与前一条同类型的历史日志消息进行差分计算；由于日志消息具有时间域的局部相似性，所以该差分压缩策略保证了日志压缩的时间与空间效率。

### 3.2.1 字段细粒度差分压缩

每个字段的细粒度差分压缩（FFDE, fine-grained field differential encoding）可以描述为一个五元组：*ffde*=(*fld*, *fld_type*, *diff_strgy*, *initVal*, *size*)，五元组中各个属性的定义如下。

*fld*：字段名。

*fld_type*：字段值类型，分为字符串、整数 2 大类，其中整数分为 8 位，16 位，32 位，64 位的有符号与无符号整数；浮点数转化为指数与尾数的 2 个整数表示。

*diff_strgy*：字段值差分策略，本日志压缩系统定义了 4 类差分策略，具体定义见表 2，对于不同特性的字段采用不同差分策略，也可扩展新差分策略以支持新类型字段。

**表 2 字段差分策略定义**

| 差分策略 | 说 明 |
|---|---|
| 定值（const） | 用于值始终固定不变的字段。将该字段值直接定义在模板字段中。编码时直接删除字段值，解码时通过模板可直接恢复 |
| 复制（copy） | 当前字段值（*val*）与前一条同类型日志消息的对应字段值（*val′*）相同时，直接删除当前字段值；否则保留该字段值 |
| 差值（delta） | 存储当前字段值（*val*）与前一条同类型日志消息对应字段值（*val′*）的差值（Δ）。对于整数，$\Delta=val^i-val'$；对于字符串值适用于相同长度的字符串，其 Δ 为字符末尾的不同字符子串。如对于日期时间字符串值，*val*=20150501001324, *val′*=20150501002326，则 $\Delta=val^i-val'=1224$ |
| 其他（other） | 未能使用已定义差分策略的字段值差分方法。它可以是原文；或是其他传统的压缩技术，由具体字段差分程序实现 |

*initVal*：字段的初始值。当第一条消息差分压缩时，将 *initVal* 作为历史值（*val′*）与当前字段值做差分运算。当字段差分策略为“定值”时，则 *initVal* 属性值定义为该字段的定值。

*size*：标识使用差分策略得到的字段差值使用定长编码（编码为 *size* 个字节）还是变长编码（*size*=0）。定长编码即保持原始值；变长编码采用被广泛使用的 LEB128[14]变长编码技术，即一个字节的 7 bit 表示实际数据，最高比特位标识当前字节是否为变长数据的最后一个字节（如 0 表示结束）。对于差分压缩后的字段值的取值范围固定（如 IP 地址的一个段的值范围为 0~255），且等于 1 字节或大于 4 个字节的定长值采用定长编码，反之采用变长编码。这样可以使存储空间达到最优。如一个无符号的 32 为整数值采用变长编码时，当其值的区间为[0, 127]内的任意一个值时，只需要一个字节存储表示，而不是 4 个字节。

### 3.2.2 字段差分存在位图

差分策略导致字段的值具有 3 种“值存在”状态：存在、不存在、条件存在。“存在”表示字段差分之后总是有值，如“差值”字段差分策略；“不存在”表示字段差分后总是没有值，如“定值”差分策略；“条件存在”表示字段差分后依据条件可能有值也可能无值。如字段差分策略为“复制”，当与前一条日志消息的值相同时，当前日志直接删除该值，差分后的字段无值；反之则保留该值，差分后的字段有值。因此一条差分后的日志消息所包含的字段值数目不固定。为了正确解码日志消息，一条差分压缩的日志消息需要存储一个字段值存在位图向量（FVPB, field value presence bitmap），FVPB 中的每一个比特对应日志中的一个具有“条件存在”状态的字段，置位表示字段值存在；复位表示字段值不存在。对于“存在”与“不存在”的字段值则不需要 FVPB 标示。

## 3.3 基于模板的日志数据细粒度差分压缩

结合 3.2 节与 3.3 节，本节给出基于模板的日志数据细粒度差分压缩总体架构。日志先通过模板去重，然后再对日志字段值进行细粒度的差分压缩。这两者均定义在日志差分压缩模板中，定义如下。

**定义 5**（细粒度差分模板，*template′*）。一个细粒度差分模板由字段值存在位图向量、各个字段的细粒度差分五元组、模板 ID 号、字段名与字段值连接符与字段分割符构成，即 $template'=\{ffde_{tid},\lambda,\theta,ffde_1,ffed_2,\cdots,ffed_N\}$。其中模板 ID 号也作为日志消息添加的一个普通字段（$ffde_{tid}$），使用“复制”差分策略（可减少连续同类型日志消除中模板 ID 的存储）。基于细粒度差分模板的日志差分压缩算法描述为算法 1。

字段差分压缩是当前值（*val*）与同类型的前一

条日志数据的对应字段值($val'$)执行差分压缩，因此整个日志的差分压缩过程需要维护一个差分字典，用于记录前一条同类型日志数据的字段值。算法 1 第 1 行为每个模板定义一个差分字典（$dict[k][0\cdots N]$），其初始值为对应模板的字段初始值（$initVal$）；每执行完一条日志消息的差分运算，用当前日志的字段值更新该差分字典（算法 1 第 6 行）。对于每一条日志消息，首先需要确定适合其的模板，然后根据模板对该日志消息进行细粒度差分压缩（算法 1 中的第 3~7 行）。

**算法 1**　基于细粒度差分模板的日志压缩算法

输入：日志 $log=\{log\_msg_i|i=1,2,\cdots,M\}$；差分压缩模板集合 $temp\_set=\{template'_k|k=1,2,\cdots,K\}$;其中 $template'_k=\{ffde_{tid}^k,\lambda,\theta, ffde_1^k,\cdots,ffed_N^k\}$, $ffed_j^k=(fld_j^k, fld\_type_j^k, diff\_strgy_j^k, initVal_j^k, size_j^k), j=1,2,\cdots,N\}$;

输出：差分压缩的日志数据;

1) 初始化模板字段差分字典 $dict[k][0\cdots N]=\{k, initVal_1^k,\cdots,initVal_N^k\}$;

2) 对于每个日志消息 $log\_msg_i$，执行如下操作；

3) 根据日志消息的结构查找与其相对应的模板 $template'_k$;

4) 对模板 $ID$ 号（$k$）与 $log\_msg_i$ 的每个字段值 $val_j^i$ 执行如下操作；

5) 依据 $diff\_strgy_j^k$ 的对 $val_j^i$ 与 $dict[k][j]$进行差分运算；如果当字段差分编码状态为“条件存在”且差分后有值，则 $FVPB[j']=1$;否则 $FVPB[j']=0$; /*$j'$ 为当前字段在字段值存储位图中的比特位*/;

6) 更新字段前值字典 $dict[k][j]=val_j$;

7) 输出 $FVPB$ 与各个字段差分后的值。

## 4　实验结果与分析

本节通过测试真实的网络系统日志数据，分析本文提出的支持大规模日志压缩系统的效率。主要考察其压缩率与压缩速度。

实验环境：处理器为 Intel Core I3-3240，内存 2 GB，Fedora-14 的操作系统。

由于提出的差分架构是流式工作方式，即只与前一条日志消息做差分计算，因此压缩总时延与日志的规模成正比关系，小规模数据集测试即可说明该架构的压缩速度。为了快速得到测试数据集的压缩率与压缩速度，本节从真实的天融信防火墙访问控制日志数据集中提出部分字段，组成 4 个数据集，数据集的特性描述如表 3 所示。

**表 3**　**数据集特性**

| 数据集 | 日志条数 | 大小/KB |
|---|---|---|
| fw_log(1w） | 1 万 | 607 |
| fw_log(10w) | 10 万 | 6 071 |
| fw_log_(20w) | 20 万 | 12 143 |
| fw_log(150w) | 150 万 | 94 831 |

对这 4 个数据集分别使用本文提出的基于模板的细粒度差分压缩架构与通用的压缩工具 gzip 进行压缩测试，选择 gzip 进行参照是因为广泛使用的开源日志管理工具 logrotate 使用的就是 gzip 作为其日志压缩算法；而且类似的日志压缩文献[6~9]的实验对比也是与 gzip 相比较。2 个压缩工具对于各个数据集的压缩时间与压缩率分别如图 1 和图 2 所示，由于 gzip 可以设置其压缩参数典型的可以设置为快速压缩模式（速度快、低压缩率）、高压缩率模式（速度慢、高压缩率）与普通压缩压缩模式（压缩速度与压缩率介于前 2 种模式之间），所以在实际测试中对于同一数据集，分别使用 3 种不同的 gzip 模式进行压缩测试。从图 1 可知，本文提出的基于模板差分的日志压缩系统在压缩速度上比普通或快速模式的 gzip 快 2~5 倍，比高压缩率模式的 gzip 快一个数量级（图 1 中的时间纵轴为对数刻度）。

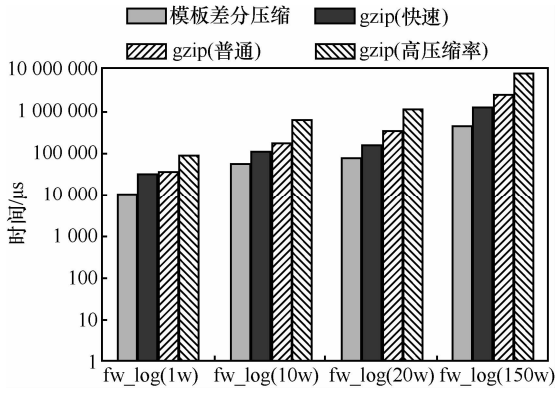


图 1　压缩时间对比

从图 2 可知，本文提出的基于模板的日志差分压缩架构的压缩率均低于 3 种模式下 gzip 的压缩率，且压缩到达为 10.5%。

图 3 描述了本文所提方法与 gzip 对于 fw_log(150w)数据集在压缩速度与压缩率 2 方面的整体对比。其中柱状图表示压缩率，黑色三角形图块表示压缩时间。图 3 表明本文提出的日志差分压缩架构与 gzip 相比，具有明显的速度优势（快 2~10 倍），同时压缩率也优于 gzip。这是合理的，因为

基于模板的差分压缩充分利用了日志数据自身的特性，在压缩前就获得了日志数据的先验知识（模板），而通用的 gzip 工具没有这些先验知识。

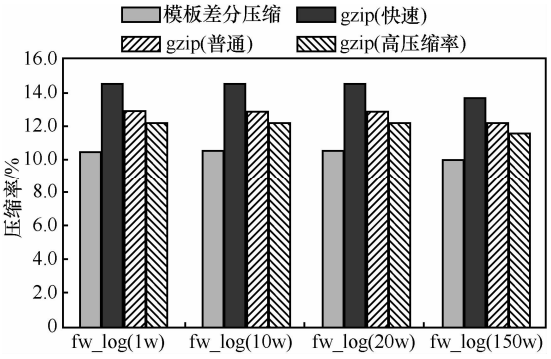


图 2　压缩率对比

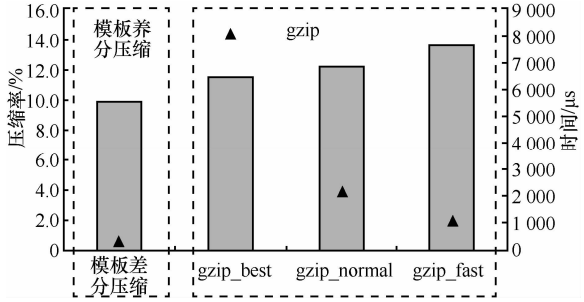


图 3　基于模板的差分压缩与 gzip 在压缩率（柱状图）与压缩速度（三角形）上的对比

## 5　结束语

基于模板流式差分的日志压缩系统首先消除日志数据中冗余的结构模式。其次，利用日志数据在时间域上的局部相似性，通过配置适用于日志内容属性（字段）的差分策略执行差分压缩，进一步降低了日志数据的存储空间。由于采用流式差分压缩，使本文所提的日志压缩架构具有显著的压缩速度优势。差分策略的可配置性使该架构具有通用性与可扩展性，该方法可以应用于一般的日志压缩。

## 参考文献：

[1] YEN T F, *et al*. Beehive: large-scale log analysis for detecting suspicious activity in enterprise networks[A]. Proceedings of the 29th Annual Computer Security Applications Conference[C]. 2013.199-208.

[2] BREIER J, BRANIŠOVÁ J. Anomaly detection from log files using data mining techniques[A]. Information Science and Applications[C]. 2015.449-457.

[3] DUMAIS S, *et al*. Understanding user behavior through log data and analysis[A]. Ways of Knowing in HCI[C]. 2014. 349-372.

[4] SRIVASTAVA M, GARG , MISHRA P K. Analysis of data extraction and data cleaning in Web usage mining[A]. Proceedings of the 2015 International Conference on Advanced Research in Computer Science Engineering Technology (ICARCSET 2015)[C]. 2015. 1-6.

[5] SKIBIŃSKI P, SWACHA J. Fast and efficient log file compression[A]. Proceedings of CEUR Workshop of 11th East-European Conference on Advances in Databases and Information Systems(ADBIS 2007)[C]. 2007.

[6] GRABOWSKI S, DEOROWICZ S. Web log compression[J]. Automatyka/Akademia Górniczo-Hutniczaim Stanisława Staszicaw Krakowie, 2007, (11): 417-424.

[7] DEOROWICZ S, GRABOWSKI S. Efficient preprocessing for Web log compression[J]. International Journal of Computing, 2008, 7(1): 35-42.

[8] DEOROWICZ S, GRABOWSKI S. Sub-atomic field processing for improved Web log compression[A]. Proceedings of IEEE International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science[C]. 2008.551-556.

[9] HÄTÖNEN K. *et al*. Comprehensive log compression with frequent patterns[A]. Data Warehousing and Knowledge Discovery[C]. 2003. 360-370.

[10] 王艳峰, 王正,阎保平. 一种高效的 DNS 日志压缩算法[J]. 计算机工程, 2010, 36(15): 32-35.
WANG Y F, WANG Z, YAN B P. High efficient DNS log compression algorithm[J]. Copular Engineering, 2010, 36(15): 32-35.

[11] CHRISTENSEN R. Improving compression of massive log data[EB/OL]. http://www.erg.utal.edu, 2013.

[12] JANG J H, *et al*. Accelerating forex trading system through transaction log compression[A]. SoC Design Conference (ISOCC), 2014 International[C]. IEEE, 2014. 24-75.

[13] LONVICK C. RFC 3164: The BSD Syslog Protocol[S]. Network Working Group.

[14] LEB128 [EB/OL]. http://en.wikipedia.org/wiki/LEB128, 2015.

## 作者简介：

**唐球**（1985-）,男，湖南怀化人，中国科学院信息工程研究所博士生，主要研究方向为正则表达式匹配、网络安全等。

**姜磊**（1984-），男，山东烟台人，博士，中国科学院信息工程研究所助理研究员，主要研究方向为正则表达式匹配、网络安全等。

**戴琼**（1975-），女，土家族，湖南慈利人，中国科学院信息工程研究所副研究员，主要研究方向为网络信息流识别与处理、网络测量与行为分析等。