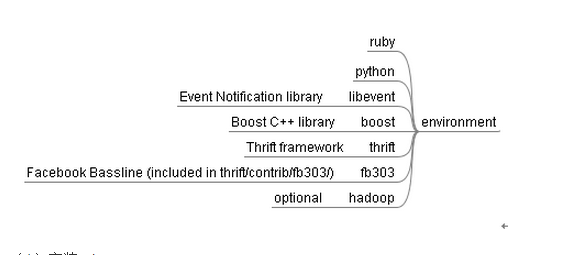
scribe日志统计系统总结

1 介绍

一个大型的分布式系统生成的日志数量是巨大的，所以需要一个有效的工具对其进行管理。Scribe就是这样一个日志收集服务器，使用它可以对大型的系统进行监控。它是Facebook的一个开源组件，使用的是Facebook另一个框架—Thrift。

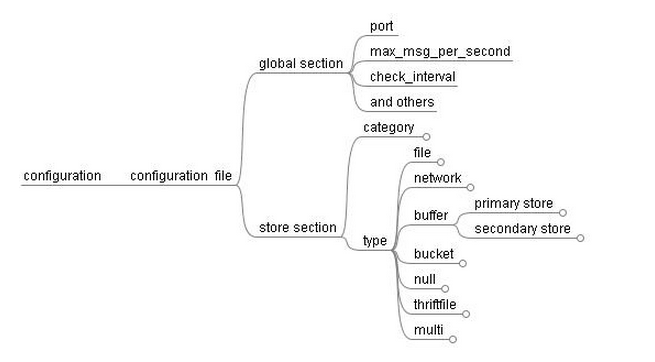
Scribe由两部分组成：central Scribe server 和 local Scribe server。在分布式系统中，每个结点都会有一个local Scribe server运行在上面，收集此结点的日志信息，并将其发送给central Scribe server。这里的central和local是相对的，是针对一个组内的。

scribe运行于linux系统，安装时有很多软件依赖性:



2 配置

Scribe server 是通过配置文件启动的，配置说明



Null : 丢弃所有的消息

File: 将消息写入文件

thriftfile：类似于file，但写入的文件类型为thrift TfileTransport文件

network: 将消息转发给另外一个scribe server

buffer: 包含主、次两个store，在主store工作时，直接使用主store记录日志，只有当主store不可用时才使用次store记录，而且一旦主store恢复工作，次store记录的日志将转移到主store中。这种方式非常适合日志的集中处理，当推到集中日志管理器失败时可以暂存在本地，当集中管理恢复后可自动将暂存保存到集中管理

bucket：包含一系列的其他类型的stroe，具体使用哪个是由所定义的Hash函数决定

multi：给所有multi类型的store传递消息

3 运行scribe server

3.1 start a scribe server:

scribed [-p <port number>] [-c <configure file>]

3.2 stop a scribe server:

scribe\_ctrl stop <port number>

3.3 use scribe\_ctrl script to monitor scribe:

scribe\_ctrl (command) <port number>

scribe\_ctrl status <port number>

scribe\_ctrl counter <port number>

4

**一.scribe配置参数的两种方式:**

1) 通过命令行,-c commandname

2) 通过指定配置文件

**二.全局参数**

1)**port**: (number)

scribe监听的端口

默认为0

可以通过命令行-p指定

2)**max\_msg\_per\_second**: (number)

每秒最大日志并发数

默认为0,0则表示没有限制

在scribeHandler::throttleDeny中使用

3)**max\_queue\_site**:(byte)

队列最大可以为多少

默认为5,000,000 bytes

在scribeHandler::Log中使用

4)**check\_interval**:(second)

检查存储的频率

默认为5

5)**new\_thread\_per\_category**:(yes/no)

是否为每个一个分类创建一个线程,为false的话,只创建一个线程为每个存储服务

默认为yes

6)**num\_thrift\_server\_threads**:(number)

接收消息的线程数

默认为3

**三.store大概配置**

1.三种store方式:

1)default store:默认分类,处理其他store无法处理的分类.每个消息都应该指定一个store cateogry

2)prefix store: 前缀分类

3)multiple categories:多个分类

2.store配置参数说明:(string)

1)**cateogry**:

哪些消息由这个category的store处理

2)**type**:

store类型,有file,buffer,network,bucket,thriftfile,null,multi

3)**target\_write\_size**:(byte)

对应category的消息在处理之前,消息队列最大可以为多大

默认为16,384

4)**max\_batch\_size**:(byte)

内存存储队列一次性可以处理的消息数量,超过这个大小将调用thrift

5)**max\_write\_interval**:(second)

对应category的消息队列在处理消息这些消息之前可以维护多长时间

默认为1秒

6)**must\_succeed**:(yes/no)

消息是否必须要成功处理,如果一个消息存储失败再重试,如果设置为no,则如果一个消息存储失败,则该条消息会被抛弃.强烈建议使用buffer store去指定一个secondary store去处理失败的消息

默认为yes

3.例子:

<store>

category=statistics

type=file

target\_write\_size=20480

max\_write\_interval=2

</store>

**四.file-store配置**

1.概述

store消息到文件

2.参数说明

1)**file\_path**:(string)

文件路径

默认为default

2)**base\_file\_name**:(string)

文件名字

默认为category名字

3)**use\_hostname\_sub\_directory**:(yes/no)

是否使用服务器的hostname作为子目录

默认为no

4)**sub\_directory**:(string)

子目录名

5)**rotate\_period**:(hourly,daily,never,number[suffix])

多长时间创建一个文件

i.hourly:多少小时;

ii.daily:多少天;

iii.never:从不;

iv.number[suffix]:其中suffix可以为s,m,h,d,w,对应秒,分钟,小时,天,星期,默认为s

默认为never

6)**rotate\_hour**:(0-23)

如果rotate\_period=daily,每隔1天多少小时创建一个文件

默认为1

7)**rotate\_minute**:(0-59)

如果rotate\_period=daily或hourly,每隔一天多少分钟或者1小时多少分钟创建一个文件

默认为15

8)**max\_site**:(bytes)

文件大约到多大时写入到一个新的文件

默认为1,000,000,000

9)**write\_meta**:(yes/no)

是否写入元数据,如果是yes,则一个文件的最后一行为write\_meta加下一个文件名

10)**fs\_type**:(std/hdfs)

文件系统类型,有std和hdfs

默认为std

11)**chunk\_size**:(number)

chunk大小,如果指定了则文件内的任何消息都不会越过这个数值,除非消息本身比chunk大

默认为0

12)**add\_newlines**:(0/1)

是否每写入一个消息就新增一行,1表示新增

默认为0

13)**create\_symlink**:(yes/no)

如果为yes,则会维护一个符号链接指向最近写入的文件

默认为yes

14)**write\_stats**:(yes/no)

如果为yes,则会为每一个store创建一个scribe\_stats文件去跟踪文件写

默认为yes

15)**max\_write\_size**:(byte)

当块大小大到max\_write\_size时,store会将数据刷新到文件系统,max\_write\_size不能超过max\_site.由于target\_write\_size大小的消息被缓存,file-store会被调用去保存这些缓存中的消息.file-store每次最少会保存max\_write\_size大小的消息,但file-store最后一次保存消息的大小肯定会小于max\_write\_size.

默认值为1,000,000

3.例子:

<store>

category=sprockets

type=file

file\_path=/tmp/sprockets

base\_filename=sprockets\_log

max\_size=1000000

add\_newlines=1

rotate\_period=daily

rotate\_hour=0

rotate\_minute=10

max\_write\_size=4096

</store>

**五.network-store配置**

1.概述

scribe可以将消息发送到其他scribe.scribe会保持长连接到其他scribe,除非发生错误或者超载才会重新连接.scribe以批处理的方式将消息发送到其他scribe

2.参数

1)**remote\_host**:(String)

要发送到的scribe服务器的host name或者ip

2)**remote\_port**:(number)

要发送到的scribe服务器的端口

3)**timeout**:(millisecond)

socket超时时间

默认为5000,对应DEFAULT\_SOCKET\_TIMEOUT\_MS变量

4)**use\_conn\_pool**:(yes/no)

是否使用连接池

默认为false

3.例子:

<store>

category=default

type=network

remote\_host=hal

remote\_port=1465

</store>

**六.buffer-store配置**

1.概述:

每个buffer-store都应该要有primary和secondary两个子store.buffer-store会先尝试将消息写到primary-store,如果写入不成功则会暂时写到secondary-store,但一旦primary-store重新接,buffer-store则又会从secondary-store读取消息再发送到primary-store,但如果replay\_buffer=no则不会这样做.secondary-store只支持这两种sotre:file和null

2.参数:

1)**buffer\_send\_rate**:(number)

每次check\_interval,做多少次从secondary-store将数据发送到primary-store

默认为1

2)**retry\_interval**:(second)

将secondary-store数据发送到primary-store的间隔,单位为秒

默认为300

3)**retry\_interval\_range**:(second)

在retry\_interval范围内随机产生一个时间间隔

默认为60

4)**replay\_buffer**:(yes/no)

如果设置为yes,会将失败的消息从secondary-store移到primary-store中

3.例子:

<store>

category=default

type=buffer

buffer\_send\_rate=1

retry\_interval=30

retry\_interval\_range=10

   <primary>

     type=network

     remote\_host=wopr

     remote\_port=1456

   </primary>

   <secondary>

     type=file

     file\_path=/tmp

     base\_filename=thisisoverwritten

     max\_size=10000000

   </secondary>

</store>

**七.bucket-store配置**

1.概述:

bucket-store可以理解为并行store,会通过每一个消息的前缀作为key散列之后写到多个文件.你可以隐式(只使用一个bucket定义)或显式的定义bucket(每个bucket使用一个bucket定义).隐式定义的bucket必须有一个名为bucket的子store,并且这个子store只能是file-store,network-store或者thriftfile-store.

2.参数:

1)**num\_buckets**:(number)

多少个bucket,如果消息无法hash则会放入一个编号为0的bucket

默认为1

2)**bucket\_type**:(key\_hash,key\_modulo,random)

bucket类型

3)**delimiter**(1-255的ascii代码)

第一次出现在消息前缀中的delimiter在key\_hash或key\_modulo中被当作key.random不会使用delimiter.

4)**remove\_key**:(yes/no)

如果为yes,则会删除每个消息的前缀key

默认为false

5)**bucket\_subdir**:(string)

如果是使用单个bucket定义,则每个文件的子目录名字为该值加bucket的hash编号

3.例子:

**1).通用例子**

<store>

category=bucket\_me

type=bucket

num\_buckets=5

bucket\_subdir=bucket

bucket\_type=key\_hash

delimiter=58

   <bucket>

     type=file

     fs\_type=std

     file\_path=/tmp/scribetest

     base\_filename=bucket\_me

   </bucket>

</store>

**2).单一定义bucket,你可以显式的定义每个bucket**

<store>

category=bucket\_me

type=bucket

num\_buckets=2

bucket\_type=key\_hash

   <bucket0>

     type=file

     fs\_type=std

     file\_path=/tmp/scribetest/bucket0

     base\_filename=bucket0

   </bucket0>

   <bucket1>

     ...

   </bucket1>

   <bucket2>

     ...

   </bucket2>

</store>

**3)定义network-store的bucket**

<store>

category=bucket\_me

type=bucket

num\_buckets=2

bucket\_type=random

   <bucket0>

     type=file

     fs\_type=std

     file\_path=/tmp/scribetest/bucket0

     base\_filename=bucket0

   </bucket0>

   <bucket1>

     type=network

     remote\_host=wopr

     remote\_port=1463

   </bucket1>

   <bucket2>

     type=network

     remote\_host=hal

     remote\_port=1463

   </bucket2>

</store>

**八.null-store配置**

1.概述:

null-store用于忽略指定category的消息

2.没有参数

3.例子:

<store>

category=tps\_report\*

type=null

</store>

**九.multi-store配置**

1.概述:

multi-store会将消息存储到它的多个子store中.一个multi-store有多个子store,命名为store0,store1,store2等.

2.参数:

1)**report\_success**:(all/any)

是否所有子sotre存储成功再报告为成功还是只要任何一个子sotre存储成功就回报为成功

默认为all

3.例子:

<store>

category=default

type=multi

target\_write\_size=20480

max\_write\_interval=1

   <store0>

     type=file

     file\_path=/tmp/store0

   </store0>

   <store1>

     type=file

     file\_path=/tmp/store1

   </store1>

</store>

**十.thriftfile-store配置**

1.概述:

thriftfile-store也是file-store的一种,只不过存储消息到的文件为TFileTransport文件

2.参数:

1)**file\_path**:(string)

要写入的文件路径

默认为/tmp

2)**base\_filename**:(string)

要写入的基本文件名

默认为category名字

3)**rotate\_period**:(hourly,daily,never,number[suffix])

多长时间创建一个文件

i.hourly:多少小时;

ii.daily:多少天;

iii.never:从不;

iv.number[suffix]:其中suffix可以为s,m,h,d,w,对应秒,分钟,小时,天,星期,默认为s

默认为never

4)**rotate\_hour**:(0-23)

如果rotate\_period=daily,每隔1天多少小时创建一个文件

默认为1

5)**rotate\_minute**:(0-59)

如果rotate\_period=daily或hourly,每隔一天多少分钟或者1小时多少分钟创建一个文件

默认为15

6)**max\_site**:(bytes)

文件大约到多大时写入到一个新的文件

默认为1,000,000,000

7)**fs\_type**:(std/hdfs)

文件系统类型,有std和hdfs

默认为std

8)**chunk\_size**:(number)

chunk大小,如果指定了则文件内的任何消息都不会越过这个数值,除非消息本身比chunk大

默认为0

9)**create\_symlink**:(yes/no)

如果为yes,则会维护一个符号链接指向最近写入的文件

默认为yes

10)**flush\_frequency\_ms**: (milliseconds)

多长时间同步thrift文件到硬盘

默认为3000

11)**msg\_buffer\_site**: (buffer)

store将会拒绝存储大于msg\_buffer\_site

默认为0,存储任何文件

3.例子:

<store>

category=sprockets

type=thriftfile

file\_path=/tmp/sprockets

base\_filename=sprockets\_log

max\_size=1000000

flush\_frequency\_ms=2000

</store>