

## Flume-ng与Mysql整合开发

我们知道，Flume可以和许多的系统进行整合，包括了Hadoop、Spark、Kafka、Hbase等等；当然，强悍的Flume也是可以和Mysql进行整合，将分析好的日志存储到Mysql（当然，你也可以存放到pg、oracle等等关系型数据库）。

不过我这里想多说一些：Flume是分布式收集日志的系统；既然都分布式了，数据量应该很大，为什么你要将Flume分析出来的数据用Mysql进行储存？能否在下面评论处留下你的使用场景呢？



如果想及时了解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog\_hadoop

其实，Flume和Mysql进行整合开发的过程也是相当的简单的。代码如下：

```
package com.iteblog.flume;

/**
 * User: 过往记忆
 * Date: 14-9-4
 * Time: 下午13:16
 * bolg:
 * 本文地址 : /archives/1109
 * 过往记忆博客，专注于hadoop、hive、spark、shark、flume的技术博客，大量的干货
 * 过往记忆博客微信公共帐号 : iteblog_hadoop
 */

import com.google.common.base.Preconditions;
import com.google.common.base.Throwables;
import com.google.common.collect.Lists;
import org.apache.flume.*;
import org.apache.flume.conf.Configurable;
```

```
import org.apache.flume.sink.AbstractSink;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.List;

public class MysqlSink extends AbstractSink implements Configurable {

    private Logger LOG = LoggerFactory.getLogger(MysqlSink.class);
    private String hostname;
    private String port;
    private String databaseName;
    private String tableName;
    private String user;
    private String password;
    private PreparedStatement preparedStatement;
    private Connection conn;
    private int batchSize;

    public MysqlSink() {
        LOG.info("MysqlSink start...");
    }

    @Override
    public void configure(Context context) {
        hostname = context.getString("hostname");
        Preconditions.checkNotNull(hostname, "hostname must be set!!!");
        port = context.getString("port");
        Preconditions.checkNotNull(port, "port must be set!!!");
        databaseName = context.getString("databaseName");
        Preconditions.checkNotNull(databaseName, "databaseName must be set!!!");
        tableName = context.getString("tableName");
        Preconditions.checkNotNull(tableName, "tableName must be set!!!");
        user = context.getString("user");
        Preconditions.checkNotNull(user, "user must be set!!!");
        password = context.getString("password");
        Preconditions.checkNotNull(password, "password must be set!!!");
        batchSize = context.getInteger("batchSize", 100);
        Preconditions.checkNotNull(batchSize > 0, "batchSize must be a positive number!!!");
    }

    @Override
```

```
public void start() {  
    super.start();  
    try {  
        //调用Class.forName()方法加载驱动程序  
        Class.forName("com.mysql.jdbc.Driver");  
    } catch (ClassNotFoundException e) {  
        e.printStackTrace();  
    }  
  
    String url = "jdbc:mysql://" + hostname + ":" + port + "/" + databaseName;  
    //调用DriverManager对象的getConnection()方法，获得一个Connection对象  
  
    try {  
        conn = DriverManager.getConnection(url, user, password);  
        conn.setAutoCommit(false);  
        //创建一个Statement对象  
        preparedStatement = conn.prepareStatement("insert into " + tableName +  
            " (content) values (?)");  
  
    } catch (SQLException e) {  
        e.printStackTrace();  
        System.exit(1);  
    }  
  
}  
  
@Override  
public void stop() {  
    super.stop();  
    if (preparedStatement != null) {  
        try {  
            preparedStatement.close();  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
  
    if (conn != null) {  
        try {  
            conn.close();  
        } catch (SQLException e) {  
            e.printStackTrace();  
        }  
    }  
}
```

```
@Override
public Status process() throws EventDeliveryException {
    Status result = Status.READY;
    Channel channel = getChannel();
    Transaction transaction = channel.getTransaction();
    Event event;
    String content;

    List<String> actions = Lists.newArrayList();
    transaction.begin();
    try {
        for (int i = 0; i < batchSize; i++) {
            event = channel.take();
            if (event != null) {
                content = new String(event.getBody());
                actions.add(content);
            } else {
                result = Status.BACKOFF;
                break;
            }
        }
    }

    if (actions.size() > 0) {
        preparedStatement.clearBatch();
        for (String temp : actions) {
            preparedStatement.setString(1, temp);
            preparedStatement.addBatch();
        }
        preparedStatement.executeBatch();

        conn.commit();
    }
    transaction.commit();
} catch (Throwable e) {
    try {
        transaction.rollback();
    } catch (Exception e2) {
        LOG.error("Exception in rollback. Rollback might not have been" +
                 "successful.", e2);
    }
    LOG.error("Failed to commit transaction." +
              "Transaction rolled back.", e);
    Throwables.propagate(e);
} finally {
    transaction.close();
}
```

```
    return result;
}
}
```

pom文件中的依赖：

```
<dependencies>
    <dependency>
        <groupId>org.apache.flume</groupId>
        <artifactId>flume-ng-core</artifactId>
    </dependency>

    <dependency>
        <groupId>org.apache.flume</groupId>
        <artifactId>flume-ng-configuration</artifactId>
    </dependency>

    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>5.1.25</version>
    </dependency>

    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-api</artifactId>
    </dependency>

    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-log4j12</artifactId>
        <scope>test</scope>
    </dependency>
</dependencies>
```

运行程序时，先在Mysql中创建一个表

```
mysql> create table mysqltest(
-> id int(11) NOT NULL AUTO_INCREMENT,
```

```
-> content varchar(50000) NOT NULL,  
-> PRIMARY KEY (`id`)  
-> ) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;  
Query OK, 0 rows affected, 1 warning (0.05 sec)
```

然后在flume中创建以下配置

```
# User: 过往记忆  
# Date: 14-9-4  
# Time: 下午13:16  
# bolg:  
# 本文地址 : /archives/1109  
# 过往记忆博客 , 专注于hadoop、hive、spark、shark、flume的技术博客 , 大量的干货  
# 过往记忆博客微信公共帐号 : iteblog_hadoop
```

```
agent.sinks.mysqlSink.type = com.iteblog.flume.MysqlSink  
agent.sinks.mysqlSink.hostname=localhost  
agent.sinks.mysqlSink.port=3306  
agent.sinks.mysqlSink.databaseName=ngmonitor  
agent.sinks.mysqlSink.tableName=mysqltest  
agent.sinks.mysqlSink.user=root  
agent.sinks.mysqlSink.password=123456  
agent.sinks.mysqlSink.channel = c1
```

用下面的命令就可以启动 :

```
bin/flume-ng agent -c conf/ -f conf/mysql_test.conf -n agent
```

再看下Mysql中的情况 :

```
mysql> select count(*) from mysqltest;  
+-----+  
| count(*) |  
+-----+  
| 98300 |  
+-----+
```

好了，开发完成！上面的程序还可以改进，可以用Mybatis进行编写，将Flume处理逻辑和业务的处理逻辑分离开，这样下次只需要处理业务，Flume那块都不需要我们去考虑了，大大降低了编程的难度。具体怎么开发我就不说了，有需要请线下讨论。

本博客文章除特别声明，全部都是原创！  
原创文章版权归过往记忆大数据（过往记忆）所有，未经许可不得转载。  
本文链接: [【】\(\)](#)