实时数据生成器

Python Socket 实时数据生成器

以下是一个使用Python编写的简单Socket服务器，可以生成实时数据并发送到指定的端口（如10050），供Flume采集：

```python
import socket
import time
import random

# 创建Socket服务器
def start_data_generator():
    host = '0.0.0.0'  # 接收所有可用IP地址
    port = 10050  # 端口号

    # 创建socket对象，指定协议和数据流类型
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((host, port))

    # 允许的最大连接数
    server_socket.listen(5)

    print(f"实时数据生成器已启动，正在监听 {port} 端口...")

    while True:
        client_socket, addr = server_socket.accept()
        print(f"连接来自: {addr}")

        try:
            while True:
                # 生成模拟订单数据
                order_id = random.randint(1000, 9999)
                product_id = random.randint(1, 100)
                quantity = random.randint(1, 10)
                price = round(random.uniform(10.0, 100.0), 2)
                timestamp = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())

                # 生成数据格式，例如：order_id, product_id, quantity, price, timestamp
                data = f"{order_id},{product_id},{quantity},{price},{timestamp}\n"

                # 向客户端发送数据
                client_socket.sendall(data.encode('utf-8'))
                print(f"发送数据: {data.strip()}")

                # 模拟延迟
                time.sleep(1)

        except BrokenPipeError:
            print(f"客户端 {addr} 已断开连接。")
```

```
44            finally:
45                client_socket.close()
46
47    # 启动数据生成器
48    if __name__ == "__main__":
49        start_data_generator()
```

说明：

1. 该脚本将启动一个Socket服务器，监听端口10050。
2. 每秒生成一条模拟订单数据，格式为 `order_id, product_id, quantity, price, timestamp` ，并发送给连接的客户端。
3. 运行此脚本后，Flume可以通过配置一个 `NetcatSource` 来连接并接收数据。

可以根据需要调整数据生成的格式和字段。

Scala版本的实时数据生成器

以下是一个使用Scala编写的简单Socket服务器，可以生成实时数据并发送到指定的端口（如10050），供Flume采集：

```scala
import java.io.PrintStream
import java.net.ServerSocket
import scala.util.Random
import java.time.LocalDateTime
import java.time.format.DateTimeFormatter
import scala.concurrent.{Future, ExecutionContext}
import scala.concurrent.ExecutionContext.Implicits.global

object RealTimeDataGenerator {

  def main(args: Array[String]): Unit = {
    val server = new ServerSocket(10050)
    println("Scala 实时数据生成器已启动，监听端口 10050...")

    while (true) {
      val socket = server.accept()
      println(s"客户端连接：${socket.getInetAddress}")

      // 使用Future处理并发连接
      Future {
        val out = new PrintStream(socket.getOutputStream)

        try {
          while (true) {
            // 生成随机订单数据
            val orderId = Random.nextInt(10000)
            val productId = Random.nextInt(100)
            val quantity = Random.nextInt(10) + 1
            val price = Random.nextDouble() * 100
            val timestamp = LocalDateTime.now.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"))

            // 拼接数据
            val data = s"$orderId,$productId,$quantity,$price,$timestamp\n"
            out.print(data)
            out.flush()

            println(s"发送数据: $data")

            // 模拟数据生成的延迟
            Thread.sleep(1000)
          }
        } catch {
          case e: Exception => println(s"连接出错：${e.getMessage}")
        } finally {
```

```
            out.close()
            socket.close()
          }
        }
      }
    }
  }
```

说明：

1. 该程序会在本地 `10050` 端口启动一个Socket服务，客户端连接后，生成并发送类似 `order_id,product_id,quantity,price,timestamp` 的订单数据。
2. 每秒生成一条数据，模拟订单系统的实时数据流。
3. 数据格式为： `订单ID, 产品ID, 购买数量, 价格, 时间戳` 。

Flume配置文件示例

假设使用Flume来采集上述Socket数据并将其发送到Kafka中的一个名为 `order` 的Topic，以下是Flume的配置文件示例：

flume.conf：

```
# Define the agent name
agent.sources = source1
agent.sinks = sink1
agent.channels = channel1

# Source configuration (using Netcat to receive socket data from the Scala generator)
agent.sources.source1.type = netcat
agent.sources.source1.bind = 0.0.0.0
agent.sources.source1.port = 10050

# Sink configuration (send data to Kafka)
agent.sinks.sink1.type = org.apache.flume.sink.kafka.KafkaSink
agent.sinks.sink1.kafka.bootstrap.servers = localhost:9092
agent.sinks.sink1.kafka.topic = order
agent.sinks.sink1.kafka.partitioner.class = org.apache.kafka.clients.producer.internals.DefaultPartitioner
agent.sinks.sink1.kafka.producer.acks = 1

# Channel configuration (using memory channel)
agent.channels.channel1.type = memory
agent.channels.channel1.capacity = 10000
agent.channels.channel1.transactionCapacity = 100

# Bind source and sink to channel
agent.sources.source1.channels = channel1
agent.sinks.sink1.channel = channel1
```

说明：

1. **Source部分**：使用 `NetcatSource` 来监听 `10050` 端口，从Scala生成器中接收数据。
2. **Sink部分**：使用 `KafkaSink` 将数据发送到Kafka中的 `order` 主题，Kafka服务器运行在 `localhost:9092`，你可以根据需要调整服务器地址和端口。
3. **Channel部分**：使用 `memory channel` 作为Flume数据流动的桥梁，确保数据从Source流向Sink。

启动Flume：

确保Flume已经安装并配置好Kafka连接，运行以下命令启动Flume：

```
flume-ng agent --conf ./conf --conf-file flume.conf --name agent -Dflume.root.logger=INFO,console
```

验证：

1. **数据发送到Kafka**：使用Kafka自带的命令行工具来查看数据：

```
kafka-console-consumer --bootstrap-server localhost:9092 --topic order --from-beginning
```

2. **多路复用备份到HDFS**：还可以配置Flume的多路复用，将数据同时写入Kafka和HDFS。

为了配置Flume将数据同时写入Kafka和HDFS，可以使用Flume的多路复用功能。通过多路复用，Flume可以将相同的数据同时发送到多个Sink，如Kafka和HDFS。

修改后的 Flume 配置文件示例：

flume-multiplex.conf：

```
# 定义 agent 名称
agent.sources = source1
agent.sinks = sink1 sink2
agent.channels = channel1 channel2

# Source 配置（Netcat 源接收来自 Scala 生成器的 Socket 数据）
agent.sources.source1.type = netcat  # 使用 netcat 作为 source 类型
agent.sources.source1.bind = 0.0.0.0  # 绑定到所有可用的网络接口
agent.sources.source1.port = 10050  # 监听端口 10050 接收数据

# Channel 选择
agent.sources.source1.channelSelector.type = replicating  # 使用复制模式，数据同时发送到多个通道
agent.sources.source1.channels = channel1 channel2  # 绑定到两个通道（channel1 和 channel2）

# Sink 1 配置（将数据发送到 Kafka）
agent.sinks.sink1.type = org.apache.flume.sink.kafka.KafkaSink  # 使用 KafkaSink 作为输出目的地
agent.sinks.sink1.kafka.bootstrap.servers = localhost:9092  # Kafka 集群的 bootstrap 服务器地址
agent.sinks.sink1.kafka.topic = order  # Kafka 主题名称为 order
agent.sinks.sink1.kafka.partitioner.class = org.apache.kafka.clients.producer.internals.DefaultPartitioner  # 使用默认的 Kafka 分区器
agent.sinks.sink1.kafka.producer.acks = 1  # 确认机制，1 表示等待 Kafka 服务器的确认

# Sink 2 配置（将数据备份到 HDFS）
agent.sinks.sink2.type = hdfs  # 使用 HDFS 作为输出目的地
agent.sinks.sink2.hdfs.path = hdfs://namenode/user/test/flumebackup/  # HDFS 备份路径
agent.sinks.sink2.hdfs.fileType = DataStream  # 数据文件类型为流式数据
agent.sinks.sink2.hdfs.writeFormat = Text  # 写入数据格式为文本格式
agent.sinks.sink2.hdfs.batchSize = 1000  # 批处理大小，1000 条事件后进行写入
agent.sinks.sink2.hdfs.rollInterval = 30  # 每隔 30 秒滚动生成一个新的文件
agent.sinks.sink2.hdfs.rollSize = 0  # 文件大小达到指定值后滚动（此处为 0，表示忽略）
agent.sinks.sink2.hdfs.rollCount = 0  # 写入指定数量的事件后滚动（此处为 0，表示忽略）

# 通道 1 配置（用于 Kafka Sink）
agent.channels.channel1.type = memory  # 使用内存通道
agent.channels.channel1.capacity = 10000  # 通道容量为 10000 条事件
agent.channels.channel1.transactionCapacity = 100  # 单次事务容量为 100 条事件

```

```
# 通道 2 配置（用于 HDFS Sink）
agent.channels.channel2.type = memory  # 使用内存通道
agent.channels.channel2.capacity = 10000  # 通道容量为 10000 条事件
agent.channels.channel2.transactionCapacity = 100  # 单次事务容量为 100 条事件

# 将 source 和 sink 绑定到各自的通道
agent.sources.source1.channels = channel1 channel2  # 将 source 绑定到两个通道
agent.sinks.sink1.channel = channel1  # 将 Kafka Sink 绑定到 channel1
agent.sinks.sink2.channel = channel2  # 将 HDFS Sink 绑定到 channel2

```

配置说明：

1. Source 部分：
   - 使用 `NetcatSource` 来监听 `10050` 端口，接收来自Scala生成器的Socket数据。
2. ChannelSelector：
   - 使用 `replicating` 类型的选择器，表示Flume会将数据同时发送到两个Channel（`channel1` 和 `channel2`），从而实现多路复用。
3. Sink 1（KafkaSink）：
   - 将数据写入Kafka的 `order` 主题。Kafka运行在 `localhost:9092`。
4. Sink 2（HDFS Sink）：
   - 将数据备份到HDFS路径 `hdfs://namenode/user/test/flumebackup/`。
   - `rollInterval=30` 表示每隔30秒会滚动生成一个新的文件。
   - `rollSize` 和 `rollCount` 设置为 `0`，表示根据时间滚动生成文件，而不是基于文件大小或事件数量。
5. Channels：
   - 为Kafka Sink和HDFS Sink分别配置了两个独立的Memory Channel。

启动 Flume：

```
flume-ng agent --conf ./conf --conf-file flume-multiplex.conf --name agent -Dflume.root.logger=INFO,console
```

验证配置：

1. **验证 Kafka 数据**：

   使用Kafka的命令行工具来验证数据是否成功发送到 `order` 主题：

```
kafka-console-consumer --bootstrap-server localhost:9092 --topic order --from-beginning
```

2. **验证 HDFS 备份**：

   可以在HDFS中查看Flume备份的文件，确认数据是否备份成功：

```
hdfs dfs -ls /user/test/flumebackup/
hdfs dfs -cat /user/test/flumebackup/<file_name>
```

通过以上配置，Flume将会将数据同时写入Kafka和HDFS。