第5章 MongoDB分片

学习目标

- 了解 MongoDB 分片
- 理解 MongoDB 分片策略
- 熟悉 MongoDB 分片集群架构
- 掌握 MongoDB 分片集群的部署
- 熟悉 MongoDB 分片的基本操作

MongoDB 分片是 MongoDB 支持的另一种集群形式，它可以满足 MongoDB 数据量呈爆发式增长的需求。当 MongoDB 存储海量的数据时，一台机器可能无法满足数据存储的需求，也可能无法提供可接受的读写吞吐量，这时，我们就可以通过在多台机器上对海量数据进行划分（即分片），使得 MongoDB 数据库系统能够存储和处理更多的数据。因此，本章我们将针对 MongoDB 分片的相关知识进行详细讲解。

5.1 分片概述

分片（sharding）技术是开发人员用来提高数据存储和数据读写吞吐量常用的技术之一。简单来说，分片主要是将数据进行划分，然后将它们分别存放于不同机器上的过程。通过使用分片可以实现降低单个机器的压力和处理更大的数据负载功能。分片与副本集主要区别在于，分片是每个结点存储数据的不同片段，而副本集是每个结点存储数据的相同副本。

所有数据库都可以进行手动分片（manual sharding），因此，分片并不是 MongoDB 特有的。不同类型的数据均可以通过人为操作被分配到不同的数据库服务器上，然而，人工分片是需要编写相关代码来实现分片功能，并且不容易维护（如集群中结点发生变动的情况）。MongoDB 数据库可以实现自动分片，它内置了多种分片逻辑，使得 MongoDB 可以自动处理分片上数据的分布，也可以很容易地管理分片集群。

数据量太大，可能导致本地磁盘不足以存储；为了提高数据库性能，从而将海量数据存储在内存中，可能导致单个 MongoDB 数据库内存不足；若出现数据请求量太大，可能导致单 MongoDB 机器不能满足读写数据的性能。若是出现这三种情况，我们就可以使用 MongoDB 的分片技术来解决。

5.2 分片策略

MongoDB 之所以能够实现自动分片，是因为其内置了分片策略。MongoDB 通过分片键(shard key)将集合中的数据划分为多个块(chunk)(默认大小为 64MB，每个块均表示集合中数据的一部分)，然后 MongoDB 根据分片策略将划分的块分发到分片集群中。注意，分片键可以是集合文档中的一个或多个字段。

MongoDB 的分片策略主要包括范围分片和哈希分片两种，具体介绍如下。

1. 范围分片(range sharding)

MongoDB 根据分片键的值范围将数据划分为不同块，每个分片都包含了分片键在一定范围内的数据。这样的话，若有文档写入时，MongoDB 会根据该文档的分片键，从而交由指定分片服务器去处理。下面，通过一张图来介绍范围分片策略，具体如图 5-1 所示。

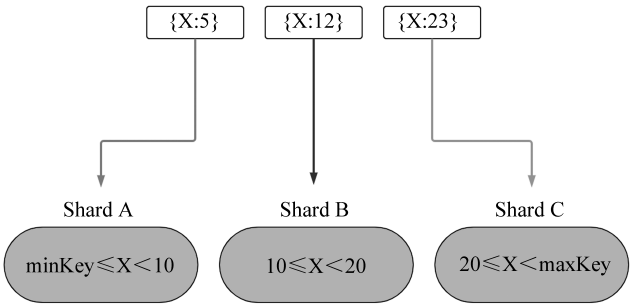

图 5-1 范围分片

从图 5-1 中可以看出，若文档分片键的值范围在[minKey,10)中，则该文档需要交由分片服务器 A 进行相关处理；若文档分片键的值范围在[10,20)中，则该文档需要交由分片服务器 B 进行相关处理；若文档分片键的值范围在[20,maxKey)中，则该文档需要交由分片服务器 C 进行相关处理。

使用基于范围分片时，拥有相近分片键的文档会存储在同一个分片服务器中，从而提升范围查询的效率。但是，当插入批量文档时，分片键集中在一定范围内，就会导致数据分布不均匀，从而导致其中一个分片服务器负载过重。

2. 哈希分片(Hash sharding)

哈希分片类似于范围分片，二者的区别在于范围分片是 MongoDB 根据分片键的值直接进行范围划分，而哈希分片则先将分片键的值进行哈希计算，然后对这些哈希值进行范围划分，从而使得每个分片都包含了哈希值在一定范围内的数据；范围分片可以支持复合分片键，而哈希分片只支持单个字段作为分片键。哈希值的随机性，使得数据随机分布在分片集群中不同的分片服务器上。下面，通过一张图来介绍哈希分片策略，具体如图 5-2 所示。

从图 5-2 中可以看出，若文档分片键的哈希值为 5，则该文档需要交由分片服务器 A 进行相关处理；若文档分片键的哈希值为 12，则该文档需要交由分片服务器 B 进行相关处理；

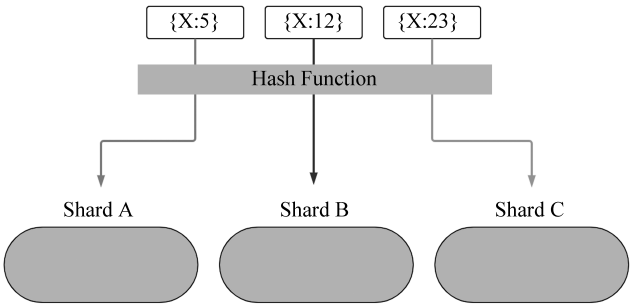

图 5-2　哈希分片

若文档分片键的哈希值为 23，则该文档需要交由分片服务器 C 进行相关处理。

使用基于哈希分片时，拥有“相近”分片键的文档不会存储在同一个分片服务器中，这样的话，数据的分离性会更好，可以保证分片集群中数据分布均衡。但是，由于数据是通过哈希计算进行随机存放的，因此会降低查询性能。

注意：

- 分片键

(1) 分片键一旦指定，后续则无法改变，并且只能拥有一个分片键。

(2) 不允许在已分片的集合文档上插入没有分片键的文档。

(3) 分片键的长度大小，不可超过 512 个字节。

(4) 用于作分片键的字段必须创建索引，索引可以是分片键开头的复合索引。

- 块(chunk)大小

(1) 小块可以均匀地分布数据，但会导致迁移很频繁，这样会增大路由服务器的开销。

(2) 大块触发的迁移较少，但会导致数据分布不均匀。

(3) 块的大小会影响要迁移块的最大文档数。

(4) 块的分片键值范围是(−∞，+∞)，其中−∞表示最小值(minKey)，+∞表示最大值(maxKey)。

5.3　分片集群架构

在 MongoDB 分片集群中，只有各组件间的协同工作，才可使得分片集群正常运行。在学习分片集群的操作之前，有必要先来学习一下分片集群架构。下面，通过一张图来介绍分片集群架构，具体如图 5-3 所示。

从图 5-3 中可以看出，分片集群中主要由三个部分组成，即分片服务器(Shard)、路由服务器(Mongos)以及配置服务器(Config Server)组成。其中，分片服务器有三个，即 Shard1、Shard2 和 Shard3；路由服务器有两个，即 Mongos1 和 Mongos2；配置服务器有三个，即主、副、副。下面，我们针对分片集群架构中的组成部分进行详细介绍，具体如下。

1. 分片服务器

分片服务器即 MongoDB 实例(即 mongod，用 Shard 表示)。分片服务器是实际存储数

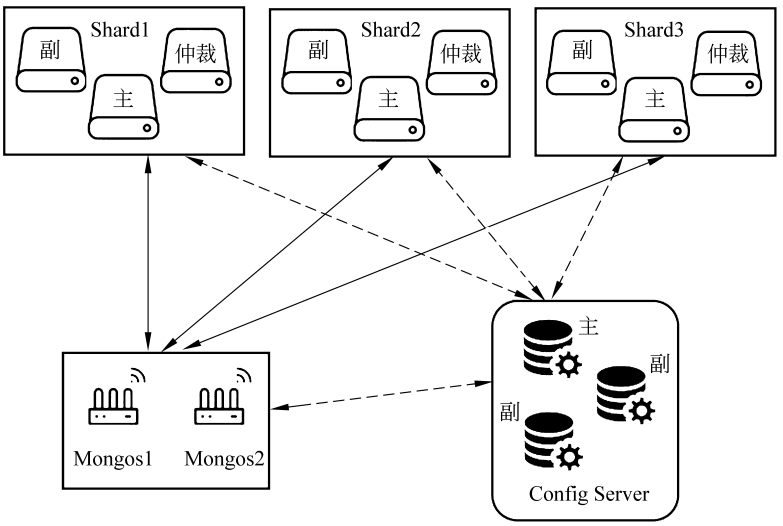

图 5-3 分片集群架构

据的组件，持有完整数据集中的一部分，每个分片服务器都可以是一个 MongoDB 实例，也可以是一组 MongoDB 实例组成的集群（副本集）。从 MongoDB 3.6 开始，必须将分片部署为副本集，这样具有更好的容错性。

2. 路由服务器

路由服务器即 mongos，主要提供客户端应用程序与分片集群交互的接口，所有请求都需要通过路由服务器进行协调工作。路由服务器实际上就是一个消息分发请求中心，它负责把客户端应用程序对应的数据请求转发到对应的分片服务器上。应用程序将查询、存储、更新等请求原封不动地发送给路由服务器。路由服务器询问配置服务器操作分片服务器需要获取哪些元数据，然后连接相应的分片服务器进行相关操作，最后将各个分片服务器的响应进行合并，返回给客户端应用程序。

生产环境中，一个分片集群通常会有多个路由服务器，一方面可以解决多个客户端同时请求，从而达到负载均衡的效果；另一方面可以解决当路由服务器宕机时导致整个分片集群无法使用的问题。

3. 配置服务器

配置服务器即 Config Server。在生产环境中，通常需要多个配置服务器，因为它存储了分片集群的元数据，并且这些数据是不允许丢失的。因此，需要配置多个配置服务器以防止数据丢失，即使其中一台配置服务器宕机，我们还有其他配置服务器，从而保证 MongoDB 分片集群依然能够正常工作。从 MongoDB 3.4 版本开始，配置服务器必须部署副本集，因此我们需要配置三个配置服务器组成的副本集。

配置服务器存储着分片集群的持久化元数据，而路由服务器存储着分片集群的非持久化元数据，这些数据均为内存缓存的数据。当路由服务器初次启动或关闭重启时，就会从配置服务器中加载分片集群的元数据。若是配置服务器的信息发生变化，则会通知所有路由

服务器更新自己的状态，这样路由服务器就能继续准确地协调客户端与分片集群的交互工作。

5.4　部署分片集群

从 MongoDB 3.6 版本开始，部署分片集群时，必须结合副本集使用。由于本书介绍的 MongoDB 版本是 4.2，因此将采用分片和副本集结合的模式部署分片集群。

5.4.1　环境准备

由于分片集群最优部署需要 14 台服务器，这样部署成本太高，并且配置服务器和路由服务器本身不存储真实数据，因此我们将路由服务器和配置服务器与分片服务器共用同一台服务器，即通过不同的进程端口号区分(实际工作中，则不建议这样做)。

在第 4 章 MongoDB 副本集中创建了三台虚拟机，即虚拟机 NoSQL_1、NoSQL_2 和 NoSQL_3。下面，通过一张图来介绍 MongoDB 分片集群的具体规划，如图 5-4 所示。

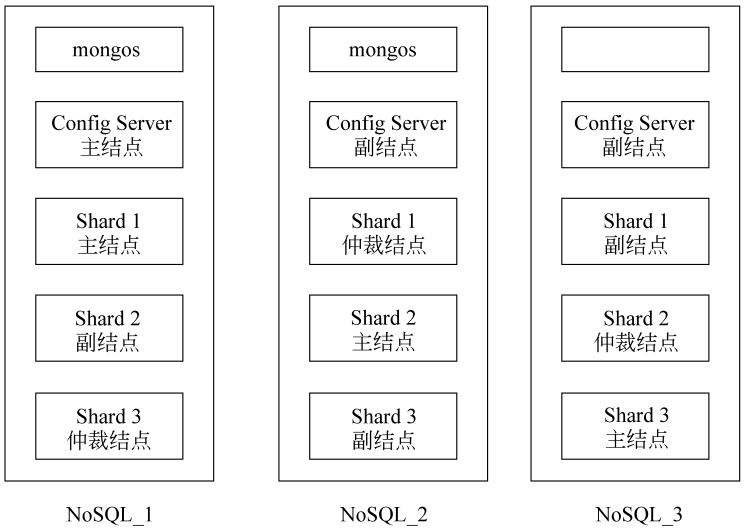

图 5-4　分片集群的规划情况

从图 5-4 中可以看出，为了保证不同虚拟机中资源的平均分配，我们在三台虚拟机中分别部署了副本集的不同结点，即虚拟机 NoSQL_1 中包含主结点 Shard1、副结点 shard2 和仲裁结点 Shard3，虚拟机 NoSQL_2 中包含仲裁结点 Shard1、主结点 Shard2 和副结点 Shard3，虚拟机 NoSQL_3 中包含副结点 Shard1、仲裁结点 Shard2 和主结点 Shard3。注意，若是在单台虚拟机中部署分片副本集的三个主结点或者三个仲裁结点，则会导致该台虚拟机负载过大或负载空闲。

由于部署分片集群时，每台虚拟机都要启动不同的服务进程，因此部署分片集群之前，需要清楚每台虚拟机已占用的端口号，从而避免出现端口冲突的情况。接下来，我们通过一张表来介绍分片集群中端口号的分配情况，如表 5-1 所示。

表 5-1 服务端口号的分配情况

虚拟机名称	服务器名称	IP 地址	Shard1	Shard2	Shard3	mongos	Config Server
NoSQL_1	nosql01	192.168.121.134	27018 主结点	27020 仲裁结点	27019 副结点	27021	27022 主结点
NoSQL_2	nosql02	192.168.121.135	27019 副结点	27018 主结点	27020 仲裁结点	27021	27022 副结点
NoSQL_3	nosql03	192.168.121.136	27020 仲裁结点	27019 副结点	27018 主结点		27022 副结点

为了规范 MongoDB 分片集群相关服务器的数据文件、配置文件以及日志文件，这里我们通过使用 user_mongo 用户分别在服务器 nosql01、nosql02 和 nosql03 的根目录下创建一些文件夹作为约定（若服务器不存在 user_mongo 用户，则参考第 3 章 3.1.2 节的内容，创建 user_mongo 用户，并授权；将目录/opt/servers/mongodb_demo/更改为用户 user_mongo 的权限），通过 mkdir -p 命令创建如下目录结构。

（1）/opt/servers/mongodb_demo/shardcluster/：存放分片集群的相关配置文件目录、日志文件目录和数据目录等内容。

（2）/opt/servers/mongodb_demo/shardcluster/configServer/configFile：存放配置服务器的配置文件。

（3）/opt/servers/mongodb_demo/shardcluster/configServer/data：存放配置服务器的数据文件。

（4）/opt/servers/mongodb_demo/shardcluster/configServer/logs：存放配置服务器的日志文件。

（5）/opt/servers/mongodb_demo/shardcluster/shard/configFile：存放分片服务器的配置文件。

（6）/opt/servers/mongodb_demo/shardcluster/shard/shard1_data：存放分片服务器 1 的数据文件。

（7）/opt/servers/mongodb_demo/shardcluster/shard/shard2_data：存放分片服务器 2 的数据文件。

（8）/opt/servers/mongodb_demo/shardcluster/shard/shard3_data：存放分片服务器 3 的数据文件。

（9）/opt/servers/mongodb_demo/shardcluster/shard/logs：存放分片服务器的日志文件。

（10）/opt/servers/mongodb_demo/shardcluster/mongos/configFile：存放路由服务器的配置文件。

（11）/opt/servers/mongodb_demo/shardcluster/mongos/logs：存放路由服务器的日志文件。

接下来，我们需要分别在三台服务器（nosql01、nosql02 和 nosql03）的配置服务器、分片服务器以及路由服务器的日志目录下，创建对应的日志管理文件。这里我们以服务器 nosql01 为例，具体如下：

```
#配置服务器日志管理文件
$touch /opt/servers/mongodb_demo/shardcluster/configServer/logs/config_server.log
#分片服务器 1 的日志管理文件
$touch /opt/servers/mongodb_demo/shardcluster/shard/logs/shard1.log
#分片服务器 2 的日志管理文件
$touch /opt/servers/mongodb_demo/shardcluster/shard/logs/shard2.log
#分片服务器 3 的日志管理文件
$touch /opt/servers/mongodb_demo/shardcluster/shard/logs/shard3.log
#路由服务器日志管理文件
$touch /opt/servers/mongodb_demo/shardcluster/mongos/logs/mongos.log
```

执行上述命令后，查看是否成功创建配置服务器、分片服务器(1、2、3)、路由服务器的日志管理文件，这里以查看服务器 nosql01 上的配置服务器日志管理文件为例进行演示，具体效果如图 5-5 所示。

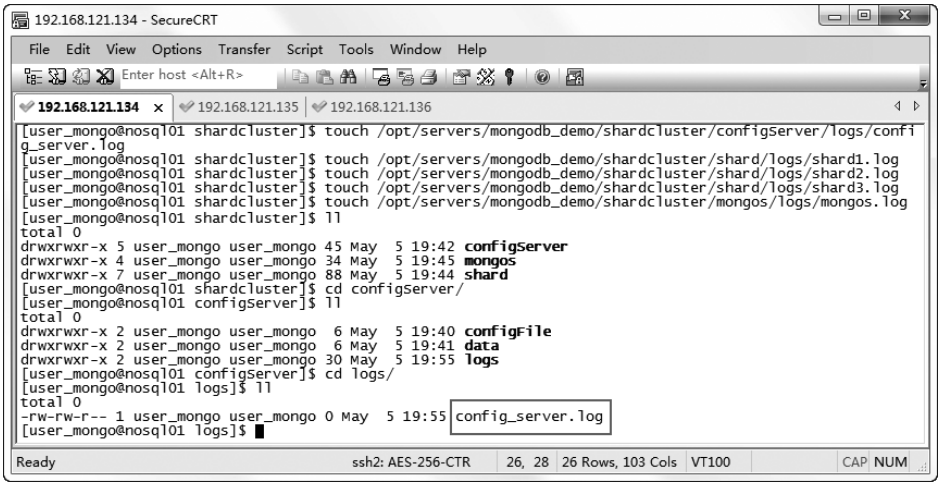

图 5-5　配置服务器的日志管理文件

从图 5-5 中可以看出，我们已经成功创建配置服务器的日志管理文件。(注：重复上述步骤，在服务器 nosql02 和服务器 nosql03 根目录下创建同样的目录结构以及日志管理文件，这里不再赘述)。至此，我们完成了 MongoDB 分片集群的环境准备工作。

脚下留心：查看端口占用情况

在分配端口号前，需要确保这些端口号不与系统中其他程序使用的端口号冲突，可在 Centos 中安装 netstat 工具查看端口占用情况，具体命令如下：

```
#安装 netstat 工具
$yum install net-tools -y
#查看端口号占用情况
$netstat -ant
```

5.4.2　部署 MongoDB

由于 MongoDB 分片集群是基于 MongoDB 的不同角色组建的，因此部署 MongoDB 分

片集群的基础仍是部署 MongoDB。部署 MongoDB 的具体步骤如下：

（1）将 MongoDB 安装包上传到服务器 nosql01 的/opt/software/目录下。

（2）将 MongoDB 安装包的用户和用户组权限修改为 user_mongo。

（3）解压安装 MongoDB，将 MongoDB 安装包解压到目录/opt/servers/mongo_demo/shardcluster/下，具体命令如下：

```
$tar -zxvf /opt/software/mongodb-linux-x86_64-rhel70-4.2.2.tgz -C /opt/
servers/mongodb_demo/shardcluster/
```

（4）解压完 MongoDB 安装包后，进入到/opt/servers/mongodb_demo/shardcluster 目录，如果觉得解压后的文件名过长，可以对文件进行重命名，具体命令如下：

```
$mv mongodb-linux-x86_64-rhel70-4.2.2/ mongodb
```

（5）将服务器 nosql01 上的 mongoDB 安装目录分发到服务器 nosql02 和 nosql03 上，具体命令如下：

```
$scp -r /opt/servers/mongodb_demo/shardcluster/mongodb user_mongo@nosql02:/opt/
servers/mongodb_demo/shardcluster/
$scp -r /opt/servers/mongodb_demo/shardcluster/mongodb user_mongo@nosql03:/opt/
servers/mongodb_demo/shardcluster/
```

执行上述命令后，需要按照提示内容连接服务器并且输入用户 user_mongo 的密码，即 123456，按照提示输入后，实现服务器 nosql01 的 mongodb 目录会分发到服务器 nosql02 和 nosql03 上，具体如图 5-6、图 5-7 和图 5-8 所示。

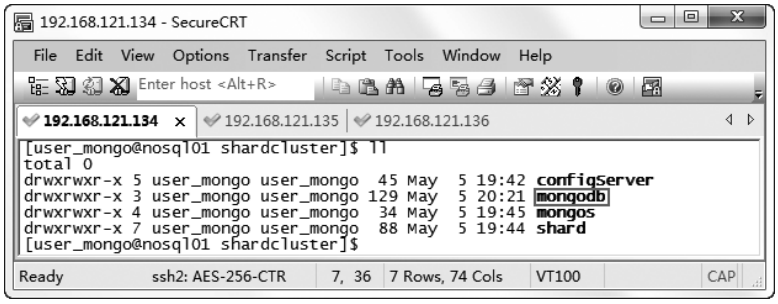

图 5-6　服务器 nosql01 的 mongodb 目录

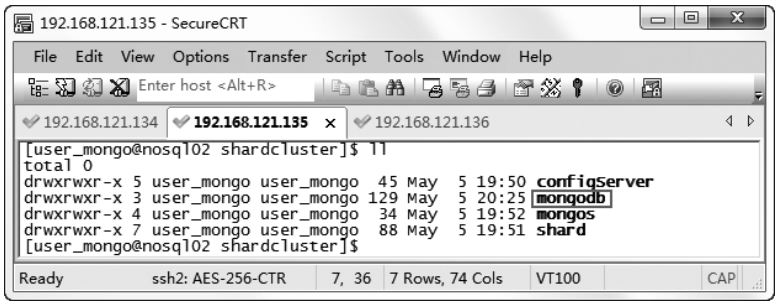

图 5-7　服务器 nosql02 的 mongodb 目录

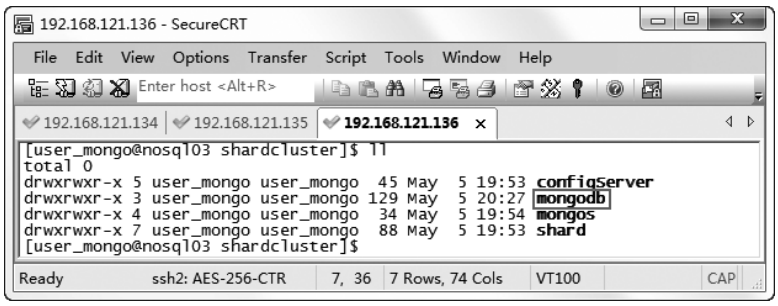

图 5-8　服务器 nosql03 的 mongodb 目录

5.4.3　部署 Config Server

上一小节完成了 MongoDB 的部署。本节我们将部署 Config Server，具体步骤如下。

1. 创建配置文件

首先，使用 user_mongo 用户在服务器 nosql01 的/configServer/configFile/目录下，创建配置文件 mongodb_config.conf，用于启动配置服务器(即 Config Server)，具体命令如下：

```
$touch /opt/servers/mongodb_demo/shardcluster/configServer/configFile/mongodb_config
.conf
```

然后，执行 vi mongodb_config.conf 命令编辑配置文件 mongodb_config.conf，添加配置服务器的相关参数，具体命令如下：

```
#数据文件存放位置
dbpath=/opt/servers/mongodb_demo/shardcluster/configServer/data
#日志文件
logpath=/opt/servers/mongodb_demo/shardcluster/configServer/logs/config_server.log
#端口号
port=27022
#绑定服务 IP
bind_ip=nosql01
#使用追加的方式写日志
logappend=true
#以守护进程的方式运行 MongoDB
fork=true
#最大同时连接数
maxConns=5000
#复制集名称
replSet=configs
#声明这是一个集群的 Config Server
configsvr=true
```

最后，将配置文件 mongodb_config.conf 通过 scp 命令分发到服务器 nosql02 和 nosql03 的目录 configFile 下，具体命令如下：

```
#分发到服务器 nosql02
$scp /opt/servers/mongodb_demo/shardcluster/configServer/configFile/mongodb_config.conf
user_mongo
@nosql02:/opt/servers/mongodb_demo/shardcluster/configServer/configFile/
#分发到服务器 nosql03
$scp /opt/servers/mongodb_demo/shardcluster/configServer/configFile/mongodb_config.conf
user_mongo
@nosql03:/opt/servers/mongodb_demo/shardcluster/configServer/configFile/
```

这里需要注意的是，参数 bind_ip 是根据当前服务器的 IP 地址或主机名进行修改的。执行上述命令后，我们必须修改服务器 nosql02 和 nosql03 配置文件 mongodb_config.conf 中的参数 bind_ip 的值，即将 bind_ip 的值修改为对应服务器的 IP 地址或主机名。

2. 启动 Config Server

分别在三台服务器（即 nosql01、nosql02 和 nosql03）MongoDB 安装目录的 bin 目录下通过配置文件方式启动 Config Server，具体命令如下：

```
$./mongod - f /opt/servers/mongodb _demo/shardcluster/configServer/configFile/mongodb _
config.conf
```

执行上述命令后，控制台会输出 Config Server 服务启动信息，若出现 successfully，则说明 Config Server 启动成功，具体如图 5-9、图 5-10 和图 5-11 所示。

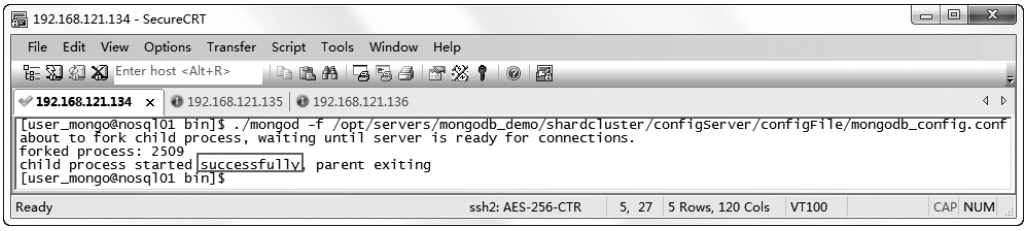

图 5-9　服务器 nosql01 中 Config Server 启动信息

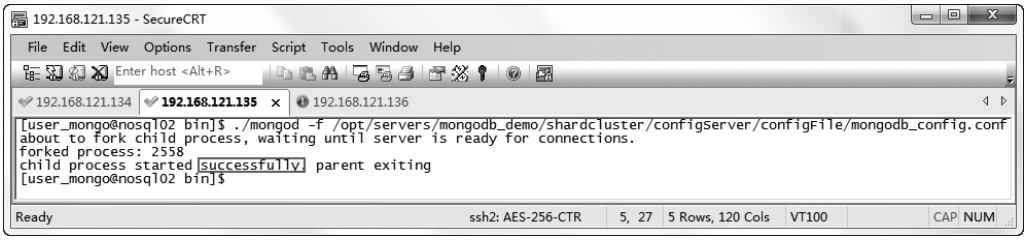

图 5-10　服务器 nosql02 中 Config Server 启动信息

从图 5-9、图 5-10 和图 5-11 中可以看出，我们已经成功启动副本集模式的 Config Server。

3. 配置 Config Server 副本集

待三台服务器的 Config Server 启动完成后，选择任意一台服务器通过 MongoDB 客户