

实验三：基于华为云安装 HBase、 Zookeeper 及 HBase 应用实践

一、实验描述

在实验一、二搭建好的集群环境上，继续安装 HBase、Zookeeper，实践 HBase 基本使用

二、实验目的

掌握 HBase、ZooKeeper 的安装与使用，使用 MapReduce 批量将 HBase 表上的数据导入到 HDFS 中，学习本实验能快速掌握 HBase 数据库在分布式计算中的应用，理解 Java API 读取 HBase 数据等相关内容。

三、实验环境

3.1 前提

确保已安装好 Hadoop 并配置好环境变量

3.2 下载安装并配置 zookeeper

在用户目录下下载 zookeeper 压缩包并解压

```
wget https://archive.apache.org/dist/zookeeper/zookeeper-3.4.6/zookeeper-3.4.6.tar.gz
```

```
mv zookeeper-3.4.6.tar.gz /usr/local
```

```
cd /usr/local
```

```
tar -zxvf zookeeper-3.4.6.tar.gz
```

建立软链接，便于后期版本更换

```
ln -s zookeeper-3.4.6 zookeeper
```

打开配置文件

```
vim /etc/profile
```

添加 ZooKeeper 到环境变量。

```
export ZOOKEEPER_HOME=/usr/local/zookeeper
```

```
export PATH=$ZOOKEEPER_HOME/bin:$PATH
```

使环境变量生效。

```
source /etc/profile
```

进入 ZooKeeper 所在目录。

```
cd /usr/local/zookeeper/conf
```

#拷贝配置文件。

```
cp zoo_sample.cfg zoo.cfg
```

修改配置文件。

```
vim zoo.cfg
```

修改数据目录。

```
dataDir=/usr/local/zookeeper/tmp
```

在最后添加如下代码

```
server.1=192.168.0.127:2888:3888
```

```
server.2=192.168.0.80:2888:3888
```

```
server.3=192.168.0.58:2888:3888
```

```
server.4=192.168.0.204:2888:3888
```

制作 tmp 目录

```
mkdir /usr/local/zookeeper/tmp
```

在 tmp 目录中创建一个空文件 myid，并向该文件写入 ID

```
touch /usr/local/zookeeper/tmp/myid
```

```
echo 1 > /usr/local/zookeeper/tmp/myid
```

配置好的 ZooKeeper 拷贝到其它节点

```
scp -r /usr/local/zookeeper-3.4.6 root@rcx-2019211279-0001:/usr/local
```

```
scp -r /usr/local/zookeeper-3.4.6 root@rcx-2019211279-0003:/usr/local
```

```
scp -r /usr/local/zookeeper-3.4.6 root@rcx-2019211279-0004:/usr/local
```

登录 rcx-2019211279-0002、rcx-2019211279-0003、rcx-2019211279-0004，创建软链接并修改 myid 内容。

rcx-2019211279-0002:

```
cd /usr/local
```

```
ln -s zookeeper-3.4.6 zookeeper
```

```
echo 2 > /usr/local/zookeeper/tmp/myid
```

rcx-2019211279-0003:

```
cd /usr/local
```

```
ln -s zookeeper-3.4.6 zookeeper
```

```
echo 3 > /usr/local/zookeeper/tmp/myid
```

rcx-2019211279-0004:

```
cd /usr/local
```

```
ln -s zookeeper-3.4.6 zookeeper
```

```
echo 4 > /usr/local/zookeeper/tmp/myid
```

分别在 rcx-2019211279-0002，rcx-2019211279-0003，rcx-2019211279-0004 上启动 ZooKeeper，需要先启动 hdfs

```
cd /usr/local/zookeeper/bin
```

```
./zkServer.sh start
```

```
./zkServer.sh status
```

查看 ZooKeeper 状态，注意，Mode 应为 leader 或 follower

3.3 下载并安装 HBase

下载 HBase:

```
wget https://archive.apache.org/dist/hbase/2.0.2/hbase-2.0.2-bin.tar.gz
```

将 hbase-2.0.2.tar.gz 放置于 rcx-2019211279-0001 节点的“/usr/local”目录，并解压。

```
mv hbase-2.0.2-bin.tar.gz /usr/local
```

```
cd /usr/local
```

```
tar -zxvf hbase-2.0.2-bin.tar.gz
```

建立软链接，便于后期版本更换。

```
ln -s hbase-2.0.2 hbase
```

编辑 “/etc/profile”文件。

```
vim /etc/profile
```

在文件底部添加环境变量

```
export HBASE_HOME=/usr/local/hbase
```

```
export PATH=$HBASE_HOME/bin:$HBASE_HOME/sbin:$PATH
```

使环境变量生效。

```
source /etc/profile
```

修改 HBase 配置文件

```
cd $HBASE_HOME/conf
```

```
vim hbase-env.sh
```

其中的 JAVA_HOME 为绝对路径

```
export JAVA_HOME=/usr/lib/jvm/jdk8u292-b10
```

```
export HBASE_MANAGES_ZK=false
```

```
export HBASE_LIBRARY_PATH=/home/modules/hadoop-2.7.7/lib/native
```

修改 hbase-site.xml 文件

```
vim hbase-site.xml
```

添加或修改 configuration 标签范围内的部分参数。

```
<configuration>
  <property>
    <name>hbase.rootdir</name>
    <value>hdfs://rcx-2019211279-0001:8020/HBase</value>
```

```

</property>
<property>
  <name>hbase.tmp.dir</name>
  <value>/usr/local/hbase/tmp</value>
</property>
<property>
  <name>hbase.cluster.distributed</name>
  <value>true</value>
</property>
<property>
  <name>hbase.unsafe.stream.capability.enforce</name>
  <value>>false</value>
</property>
<property>
  <name>hbase.zookeeper.quorum</name>
  <value>rcx-2019211279-0002:2181,rcx-2019211279-0003:2181,rcx-
2019211279-0004:2181</value>
</property>
<property>
  <name>hbase.unsafe.stream.capability.enforce</name>
  <value>>false</value>
</property>
</configuration>

```

修改 regionservers

编辑 regionservers 文件

vim regionservers

文件内容替换为 agent 节点 IP

rcx-2019211279-0002

rcx-2019211279-0003

rcx-2019211279-0004

拷贝 hdfs-site.xml

拷贝 hadoop 目录下的 hdfs-site.xml 文件到“hbase/conf”目录,可选择软链接或拷贝。

cp /home/modules/hadoop-2.7.7/etc/hadoop/hdfs-site.xml /usr/local/hbase/conf/hdfs-site.xml

拷贝 hbase-2.0.2 到 rcx-2019211279-0002、rcx-2019211279-0003、rcx-2019211279-0004 节点的 “/usr/local”目录。

```
scp -r /usr/local/hbase-2.0.2 root@rcx-2019211279-0001:/usr/local/
```

```
scp -r /usr/local/hbase-2.0.2 root@rcx-2019211279-0003:/usr/local/
```

```
scp -r /usr/local/hbase-2.0.2 root@rcx-2019211279-0004:/usr/local/
```

分别登录到 rcx-2019211279-0002、rcx-2019211279-0003、rcx-2019211279-0004 节点，为 hbase-2.0.2 建立软链接

```
cd /usr/local
```

```
ln -s hbase-2.0.2 hbase
```

依次启动 ZooKeeper 和 Hadoop。

在 rcx-2019211279-0001 节点上启动 HBase 集群。

```
/usr/local/hbase/bin/start-hbase.sh
```

观察进程是否都正常启动。

```
jps
```

3.3 HBase 实践

1 启动 Hadoop 集群

在 rcx-2019211279-0001 运行：

```
start-dfs.sh
```

```
start-yarn.sh
```

2 启动 Zookeeper 集群

需要在 rcx-2019211279-000{2..4} 分别运行：

```
./usr/local/zookeeper/bin/zkServer.sh start
```

3 启动 HBase 集群

在 rcx-2019211279-0001 运行

进入 HBase Shell 创建实验用表

输入 hbase shell 进入 hbase 交互式环境

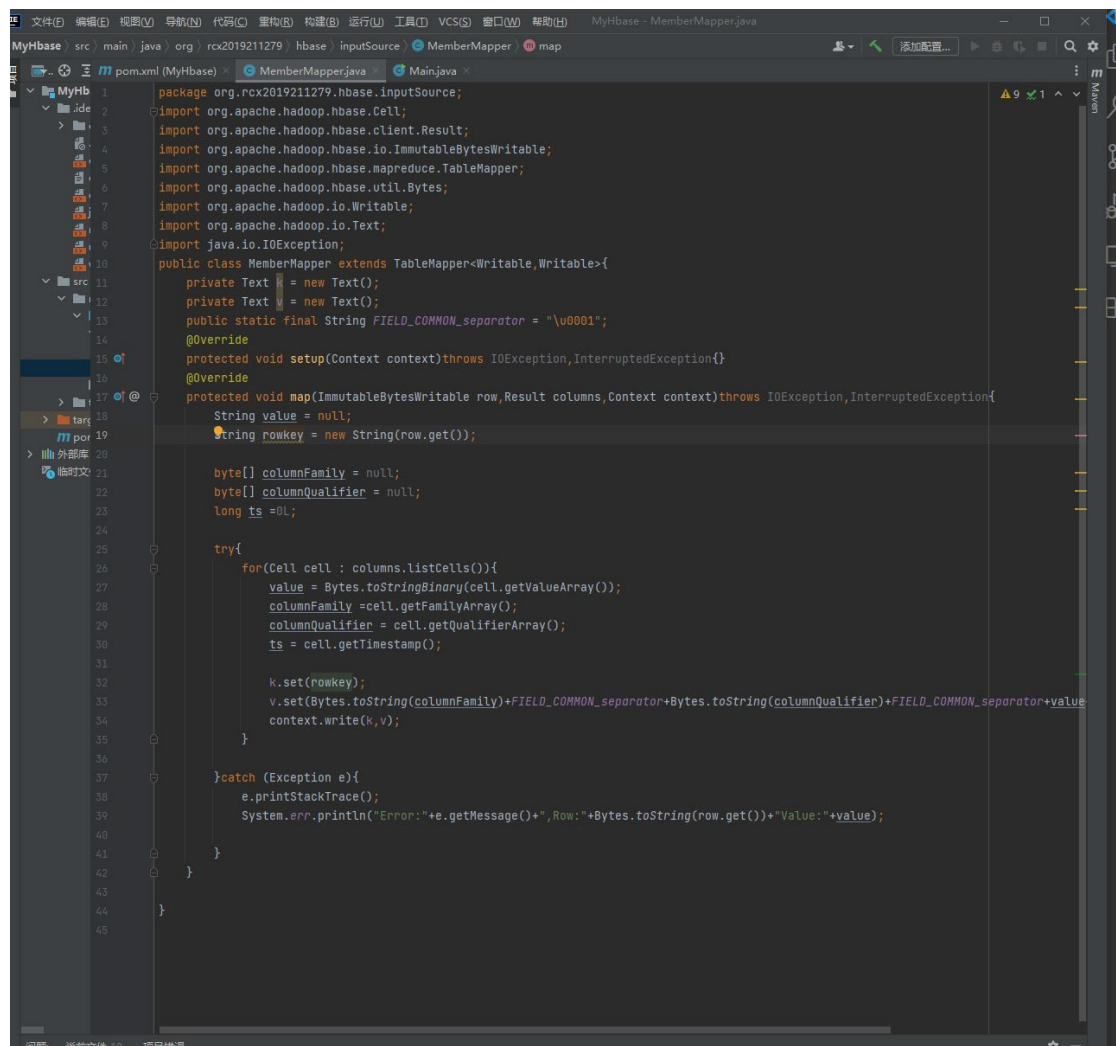
4 数据库表格设计并插入数据扫描表格：

```
cmdline.txt  hbase-sheet.txt X  hbase-site-part.xml
2022Spring > 大数据基础 > 实验三 > code_source > hbase-sheet.txt
1  hbase shell
2
3
4  create '2019211279-rcx','cf1'
5  put '2019211279-rcx','2019211279-rcx-0001','cf1:keyword','C'
6  put '2019211279-rcx','2019211279-rcx-0002','cf1:keyword','C++'
7  put '2019211279-rcx','2019211279-rcx-0003','cf1:keyword','JAVA'
8  put '2019211279-rcx','2019211279-rcx-0004','cf1:keyword','Python'
9
10 scan '2019211279-rcx'|
11
```

```
[root@rcx-2019211279-0001 local]# hbase shell
create '2019211279-rcx','cf1'
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase-2.0.2/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/modules/hadoop-2.7.7/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2022-04-30 00:52:14,250 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 2.0.2, r1cfab033e779df840d5612a85277f42a6a4e8172, Tue Aug 28 20:50:40 PDT 2018
Took 0.0086 seconds
hbase(main):001:0> create '2019211279-rcx','cf1'
Created table 2019211279-rcx
Took 1.9527 seconds
=> Hbase::Table - 2019211279-rcx
hbase(main):002:0> put '2019211279-rcx','2019211279-rcx-0001','cf1:keyword','C'
Took 0.2002 seconds
hbase(main):003:0> put '2019211279-rcx','2019211279-rcx-0002','cf1:keyword','C++'
Took 0.0070 seconds
hbase(main):004:0> put '2019211279-rcx','2019211279-rcx-0003','cf1:keyword','JAVA'
Took 0.0061 seconds
hbase(main):005:0> put '2019211279-rcx','2019211279-rcx-0004','cf1:keyword','Python'
Took 0.0282 seconds
hbase(main):006:0> scan '2019211279-rcx'
ROW COLUMN+CELL
 2019211279-rcx-0001 column=cf1:keyword, timestamp=1651251181363, value=C
 2019211279-rcx-0002 column=cf1:keyword, timestamp=1651251188513, value=C++
 2019211279-rcx-0003 column=cf1:keyword, timestamp=1651251196025, value=JAVA
 2019211279-rcx-0004 column=cf1:keyword, timestamp=1651251201805, value=Python
4 row(s)
Took 0.0555 seconds
hbase(main):007:0>
hbase(main):008:0*
hbase(main):009:0*
hbase(main):010:0*
hbase(main):011:0* □
```

5 编写代码，将 Hbase 中的数据导出到 hdfs 指定目录

打开 IDEA，新建 maven 工程，工程名 MyHBase，编写 pom.xml 文件添加依赖
在 src/java 目录下新建 package，名称 org/rcx2019211279/hbase/inputSource
新建类 MemberMapper



```
package org.rcx2019211279.hbase.inputSource;
import org.apache.hadoop.hbase.Cell;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.io.ImmutableBytesWritable;
import org.apache.hadoop.hbase.mapreduce.TableMapper;
import org.apache.hadoop.hbase.util.Bytes;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.io.Text;
import java.io.IOException;

public class MemberMapper extends TableMapper<Writable,Writable>{
    private Text k = new Text();
    private Text v = new Text();
    public static final String FIELD_COMMON_separator = "\u0001";

    @Override
    protected void setup(Context context)throws IOException,InterruptedException{}

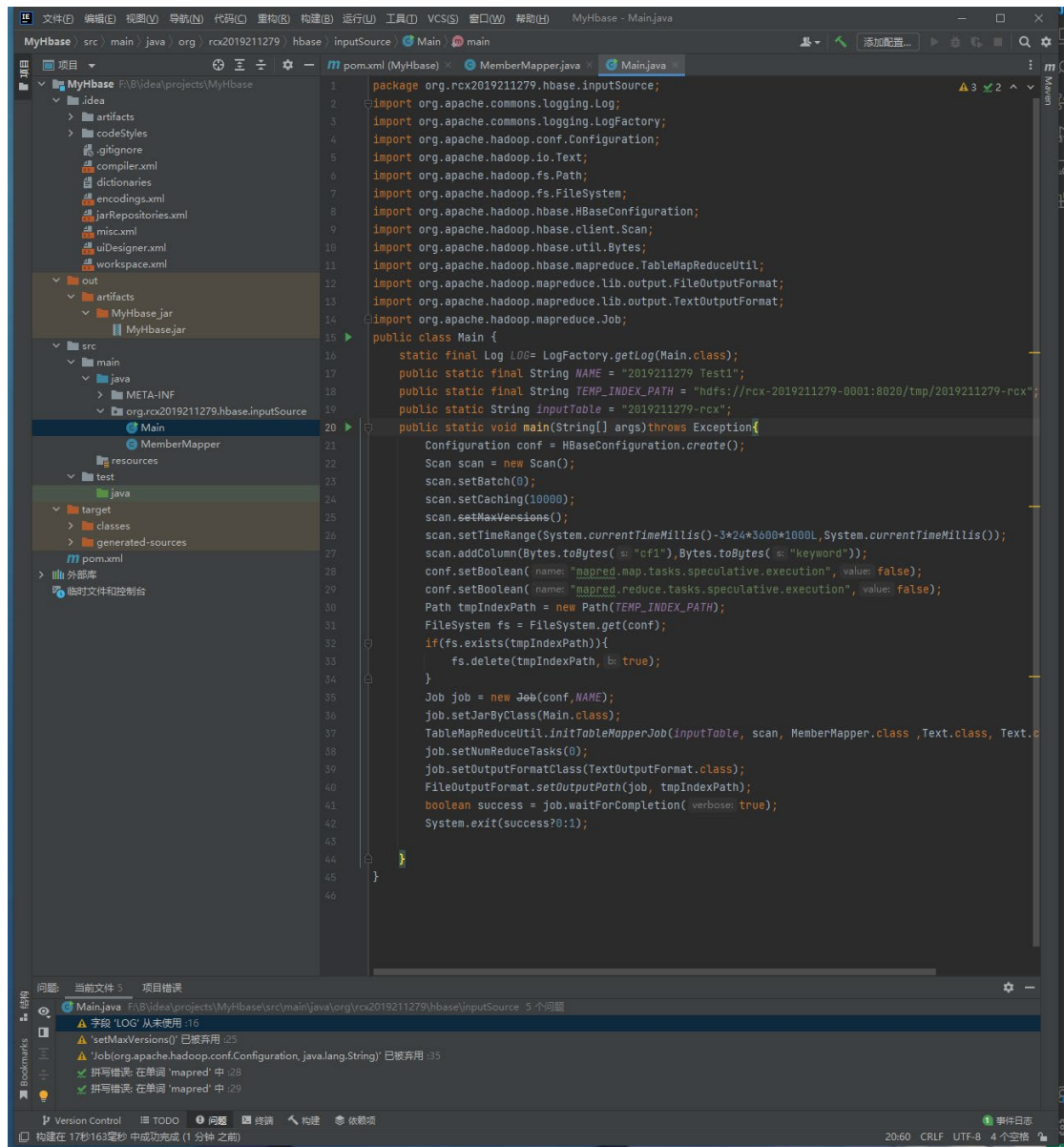
    @Override
    protected void map(ImmutableBytesWritable row,Result columns,Context context)throws IOException,InterruptedException{
        String value = null;
        String rowkey = new String(row.get());

        byte[] columnFamily = null;
        byte[] columnQualifier = null;
        long ts = 0L;

        try{
            for(Cell cell : columns.listCells()){
                value = Bytes.toStringBinary(cell.getValueArray());
                columnFamily = cell.getFamilyArray();
                columnQualifier = cell.getQualifierArray();
                ts = cell.getTimestamp();

                k.set(rowkey);
                v.set(Bytes.toString(columnFamily)+FIELD_COMMON_separator+Bytes.toString(columnQualifier)+FIELD_COMMON_separator+value);
                context.write(k,v);
            }
        }catch (Exception e){
            e.printStackTrace();
            System.err.println("Error:"+e.getMessage()+" ,Row:"+Bytes.toString(row.get())+"Value:"+value);
        }
    }
}
```

新建类 Main



6 打包程序,导出 jar 包

7 将 jar 包通过 winscp 或 scp 命令复制到服务器 rcx-2019211279-0001 上运行 jar 包

8 查看结果

[illegible]

四、实验结果与分析

4.1

```
[root@rcx-2019211279-0001 local]# hbase shell
create '2019211279-rcx','cf1'
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hbase-2.0.2/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/home/modules/hadoop-2.7.7/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2022-04-30 00:52:14,250 WARN [main] util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
HBase Shell
Use "help" to get list of supported commands.
Use "exit" to quit this interactive shell.
Version 2.0.2, r1cfab033e779df840d5612a85277f42a6a4e8172, Tue Aug 28 20:50:40 PDT 2018
Took 0.0086 seconds
hbase(main):001:0> create '2019211279-rcx','cf1'
Created table 2019211279-rcx
Took 1.9527 seconds
=> Hbase::Table - 2019211279-rcx
hbase(main):002:0> put '2019211279-rcx','2019211279-rcx-0001','cf1:keyword','C'
Took 0.2002 seconds
hbase(main):003:0> put '2019211279-rcx','2019211279-rcx-0002','cf1:keyword','C++'
Took 0.0070 seconds
hbase(main):004:0> put '2019211279-rcx','2019211279-rcx-0003','cf1:keyword','JAVA'
Took 0.0061 seconds
hbase(main):005:0> put '2019211279-rcx','2019211279-rcx-0004','cf1:keyword','Python'
Took 0.0282 seconds
hbase(main):006:0> scan '2019211279-rcx'
ROW                                COLUMN+CELL
 2019211279-rcx-0001                column=cf1:keyword, timestamp=1651251181363, value=C
 2019211279-rcx-0002                column=cf1:keyword, timestamp=1651251188513, value=C++
 2019211279-rcx-0003                column=cf1:keyword, timestamp=1651251196025, value=JAVA
 2019211279-rcx-0004                column=cf1:keyword, timestamp=1651251201805, value=Python
4 row(s)
Took 0.0555 seconds
hbase(main):007:0>
hbase(main):008:0*
hbase(main):009:0*
hbase(main):010:0*
hbase(main):011:0* □
```

截图一

五、经验总结

在本次实验中，我遇到一个问题，不小心删除某些文件夹内容导致 `hadoop` 无法正常启动，我尝试了重新格式化 `Hadoop`，但是由于忘记了删除 `tmp` 文件夹的内容，导致重新格式化后的 `Hadoop` 仍然无法正常使用，且会报错提示无可用的 `datanode`。

这个问题解决办法之前就有同学提到，删除所有节点的 `hadoop` 下的 `tmp` 文件夹即可。但是为什么会出现这个问题？

我在网上找到了一个合理的解释：提示无可用的 `datanode` 是因为多次格式化 `Hadoop` 导致 `namenode` 的 `clusterID` 与 `datanode` 的 `clusterID` 不一致，`namenode` 的 `clusterID` 在每次格式化都会更新，`datanode` 的 `clusterID` 只有在首次格式化时才会更新。而他们的 `clusterID` 分别在 `tmp` 文件夹下的 `/data/version` 和 `/name/version` 中。

所以删除所有节点的 `tmp` 文件夹是一个办法，但如果想要保留 `tmp` 文件夹的内容，那么可以将 `tmp/name/version` 中的 `clusterID` 拷贝替换到 `tmp/data/version` 中