FusionInsight

HDFS应用开发

www.huawei.com

HUAWEI

目标

- 学完本课程后，您将能够:
  - 了解**HDFS**应用开发场景
  - 掌握**HDFS**业务开发流程
  - 掌握**HDFS**常用接口使用
  - 掌握**HDFS**应用开发实践

目录

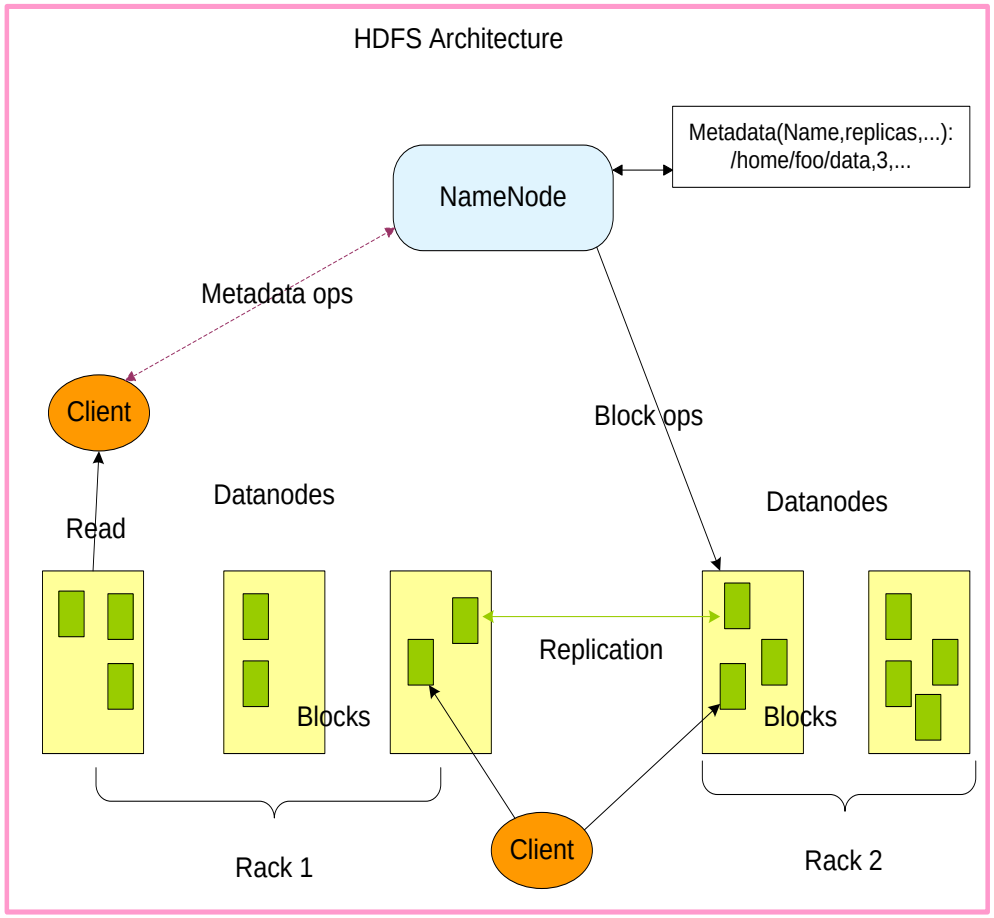
HDFS概述

- **HDFS(Hadoop Distributed File System)**基于**Google**发布的**GFS**论文设计开发，适合运行在通用硬件上的分布式文件系统。其除具备其它分布式文件系统相同特性外，还有自己特有的特性：
  - 高容错性：认为硬件总是不可靠的
  - 高吞吐量：为大量数据访问应用提供高吞吐量支持
  - 大文件存储：支持存储**TB-PB**级别的数据

HDFS适合做什么？
大文件存储、流式数据访问

HDFS不适合做什么？
大量小文件、随机写入、低延迟读取

基本系统架构



HDFS架构包含三个部分：

NameNode，DataNode，Client

- **NameNode**：**NameNode**用于存储、生成文件系统的元数据。运行一个实例。
- **DataNode**：**DataNode**用于存储实际的数据，将自己管理的数据块上报给**NameNode** ，运行多个实例。
- **Client**：支持业务访问**HDFS**，从**NameNode ,DataNode**获取数据返回给业务。多个实例，和业务一起运行。

目录

HDFS数据读取流程

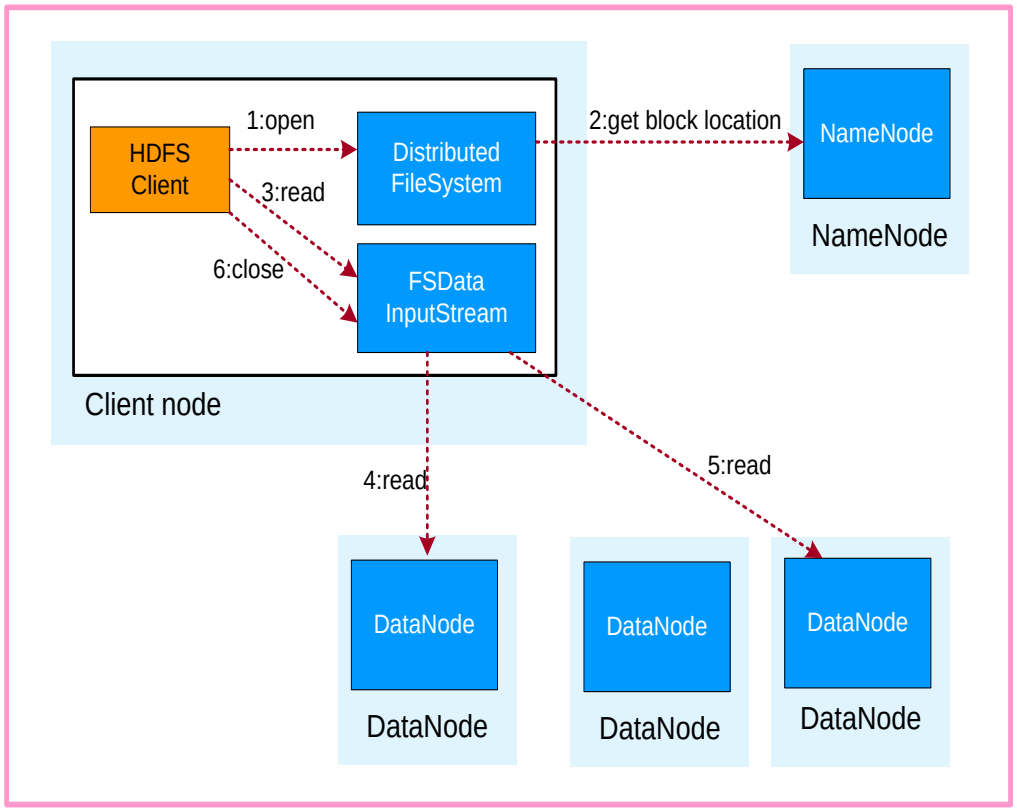
HDFS数据读取流程如下：

1. 业务应用调用**HDFS Client**提供的**API**打开文件。

2. HDFS Client联系**NameNode**，获取到文件信息（数据块、**DataNode**位置信息）。

3. 业务应用调用**read API**读取文件。

4. HDFS Client根据从**NameNode**获取到的信息，联系**DataNode**，获取相应的数据块。**(Client**采用就近原则读取数据**)**。



5. HDFS Client会与多个**DataNode**通讯获取数据块。

6. 数据读取完成后，业务调用**close**关闭连接。

HDFS数据写入流程

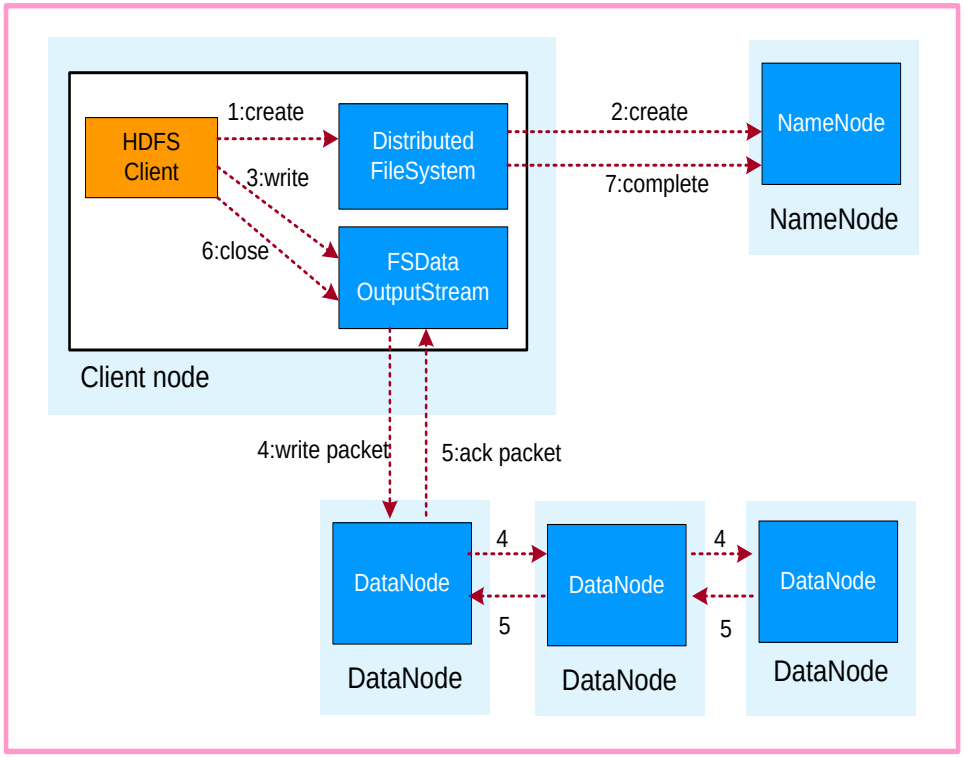
HDFS数据写入流程如下：

1. 业务应用调用**HDFS Client**提供的**API**创建文件，请求写入。

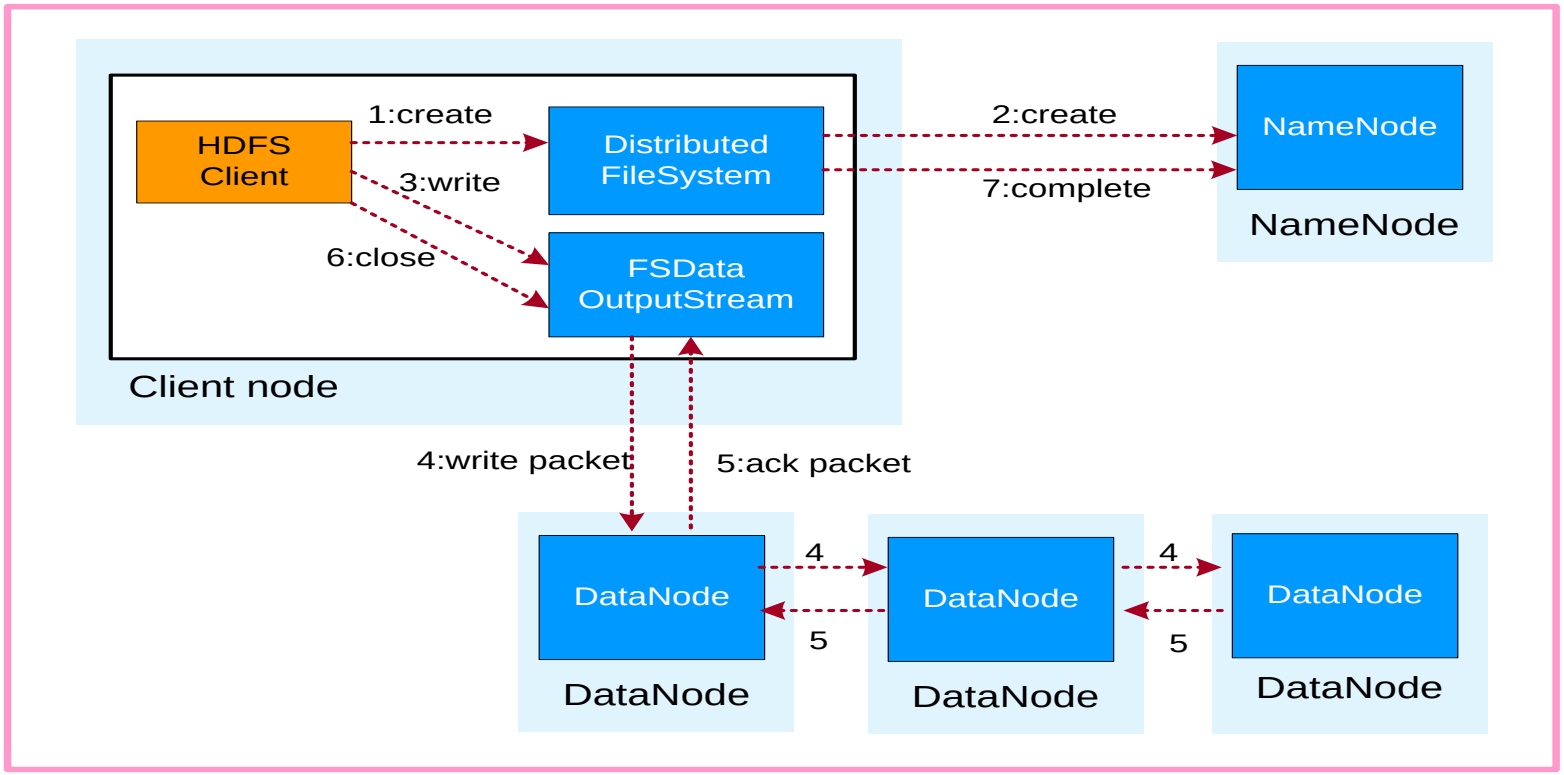
2. HDFS Client联系**NameNode**，**NameNode**在元数据中创建文件节点。

3. 业务应用调用**write API**写入文件。

4. HDFS Client收到业务数据后，从**NameNode**获取到数据块编号、位置信息后，联系**DataNode**，并将需要写入数据的**DataNode**建立起流水线。完成后，客户端再通过自有协议写入数据到**DataNode1**，再由**DataNode1**复制到**DataNode2, DataNode3**。



HDFS数据写入流程（续）



5. 写完的数据，将返回确认信息给**HDFS Client**。

6. 所有数据确认完成后，业务调用**HDFS Client**关闭文件。

7. 业务调用**close**，**flush**后**HDFS Client**联系**NameNode**，确认数据写完成，**NameNode**持久化元数据。

目录

HDFS应用开发方式

- **HDFS Client**：**HDFS**应用开发需要使用**HDFS Client**模块，**Client**主要包括三种方式。

  Java

  - 提供**HDFS**文件系统的应用接口，使用**Java API**进行**HDFS**文件系统应用开发。

  Shell

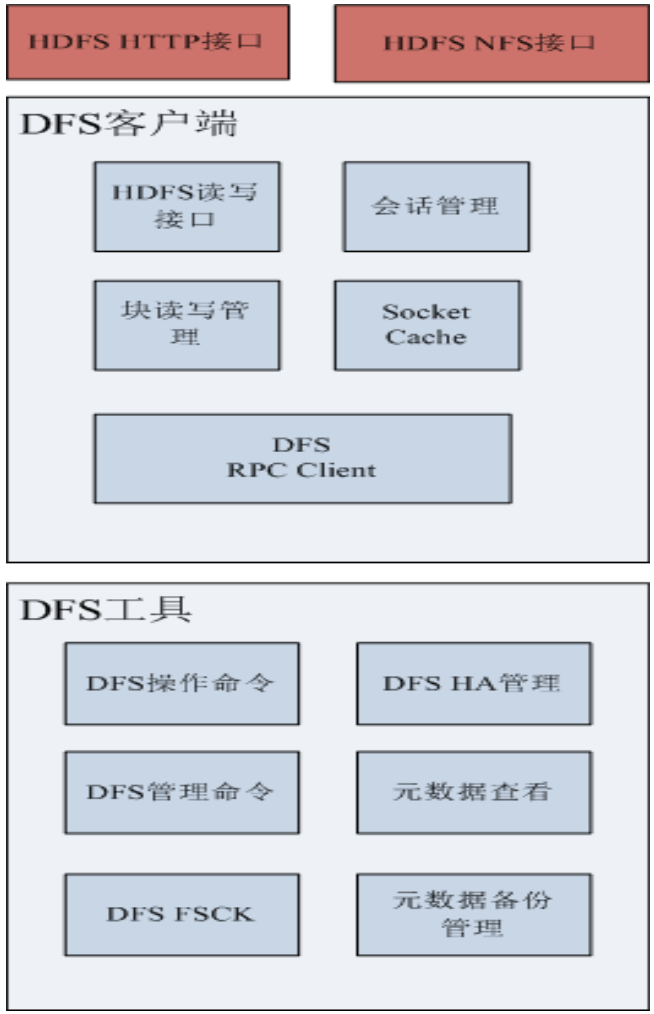
  - 提供**shell**命令完成**HDFS**文件系统的基本操作。

  Web UI

  - 提供**Web**可视化组件管理界面。

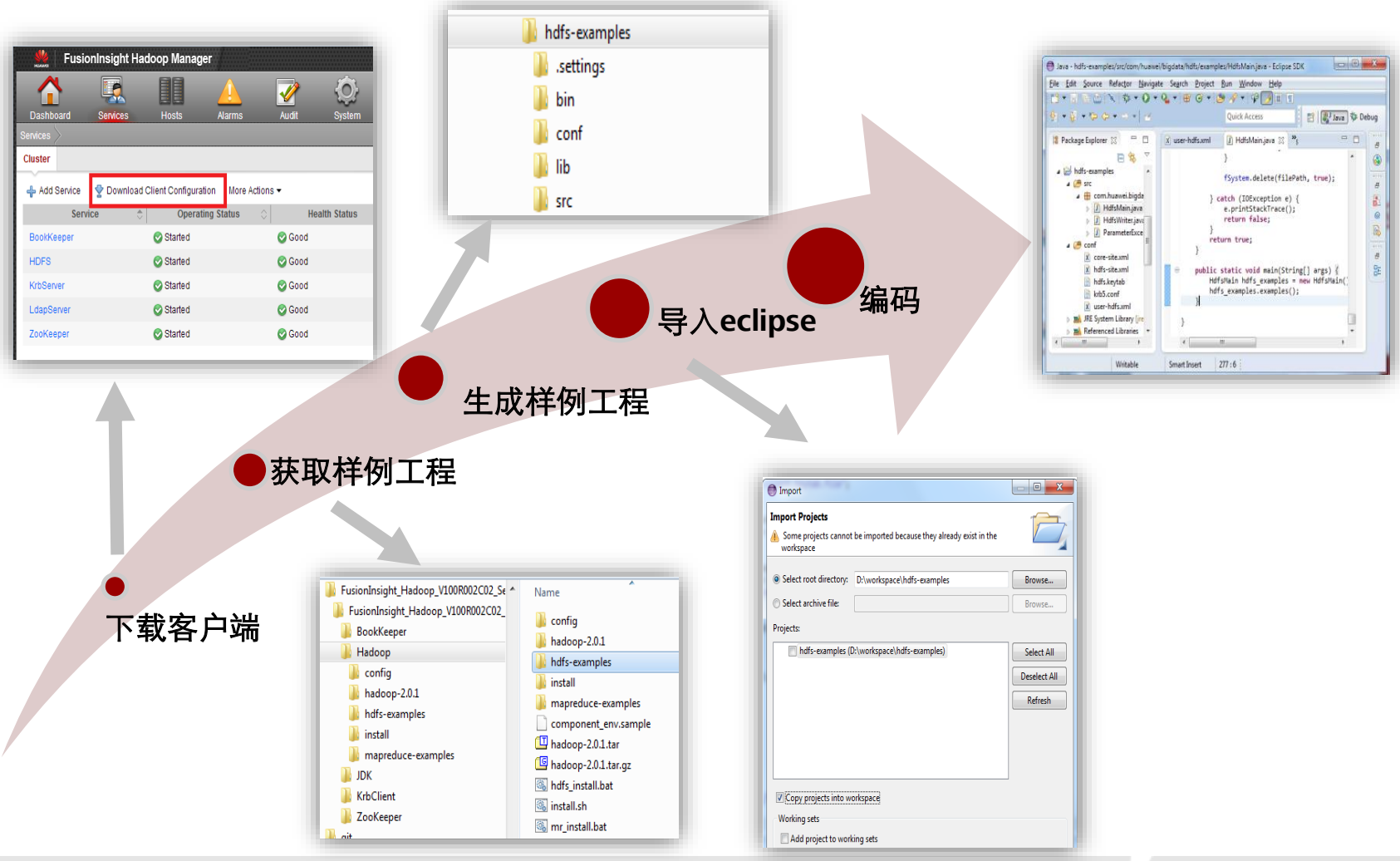
- **Kerberos**控制

  应用程序采用密钥文件在**FusionInsight Hadoop**产品中进行**Kerberos**安全认证。

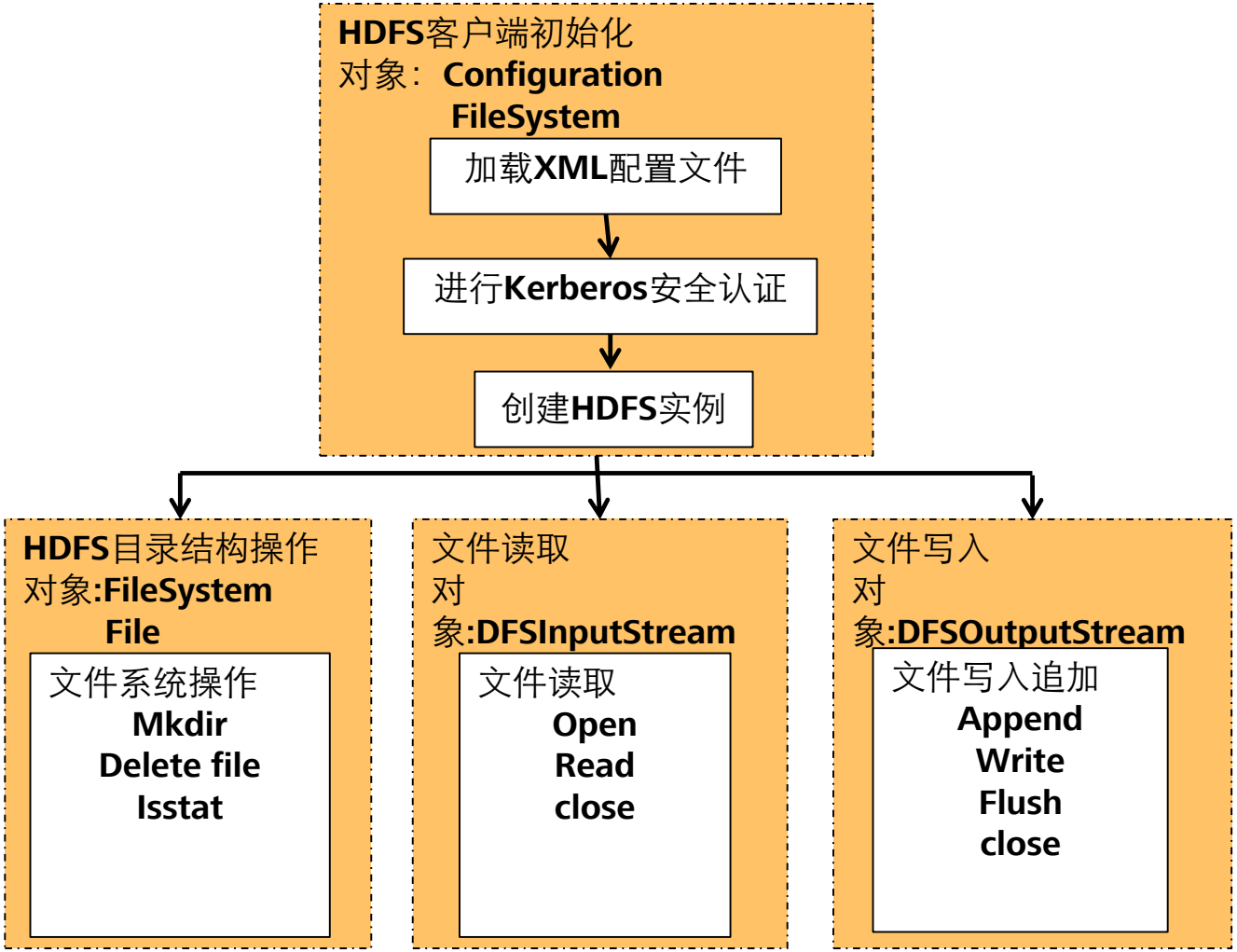


目录

开发环境准备



JAVA开发流程



开发工程示例-初始化

初始化流程：

1. 初始化配置对象，并加载配置文件。（配置文件包含访问**HDFS**的必要信息，如**IP**地址）

2. HDFS安全版本下，配置访问**HDFS**安全认证的用户以及**keytab**文件，进行安全认证和登录。

3. 安全认证登录成功后，实例化**Filesystem**对象。

```
private void confLoad() throws IOException {
  conf = new Configuration();
  // conf file
  conf.addResource(new Path(PATH_TO_HDFS_SITE_XML));
  conf.addResource(new Path(PATH_TO_CORE_SITE_XML));
  //conf.addResource(new Path(PATH_TO_SMALL_SITE_XML));
}
```

```
private void authentication() throws IOException {
  // security mode
  if ("kerberos".equalsIgnoreCase(conf.get("hadoop.security.authentication"))) {
    String PATH_TO_KEYTAB = HdfsMain.class.getClassLoader().getResource("user.keytab").getPath();
    String PATH_TO_KRB5_CONF = HdfsMain.class.getClassLoader().getResource("krb5.conf").getPath();
    System.setProperty("java.security.krb5.conf", PATH_TO_KRB5_CONF);
    LoginUtil.login(PRNCIPAL_NAME, PATH_TO_KEYTAB, PATH_TO_KRB5_CONF, conf);
  }
}
```

```
private void instanceBuild() throws IOException {
  // get filesystem
  fSystem = FileSystem.get(conf);
}
```

开发工程示例-目录操作

创建目录流程：

1. 判断目录路径是否存在；
2. 不存在时创建目录。

```
private boolean createPath(final Path filePath) throws IOException {
  if (!fSystem.exists(filePath)) {
    fSystem.mkdirs(filePath);
  }
  return true;
}
```

删除目录流程：

1. 判断目录路径是否存在；
2. 存在时删除目录。

```
private boolean deletePath(final Path filePath) throws IOException {
  if (!fSystem.exists(filePath)) {
    return false;
  }
 // fSystem.delete(filePath, true);
  return fSystem.delete(filePath, true);
}
```

涉及API：

API名称	API描述
FileSystem.exists	判断指定的目录文件是否存在
FileSystem.mkdirs	在**HDFS**中创建指定的目录
FileSystem.delete	删除目录，可递归删除

开发工程示例-文件读取

数据读取流程：

1. 初始化**FSDataInputstream**对象。

2. 使用**FSDataInputStream**对象初始化**BufferReader**。

3. 使用**BufferReader.read**读取**HDFS**数据。

```
private void read() throws IOException {
  String strPath = DEST_PATH + File.separator + FILE_NAME;
  Path path = new Path(strPath);
  FSDataInputStream in = null;
  BufferedReader reader = null;
  StringBuffer strBuffer = new StringBuffer();

  try {
    in = fSystem.open(path);
    reader = new BufferedReader(new InputStreamReader(in));
    String sTempOneLine;

    // write file
    while ((sTempOneLine = reader.readLine()) != null) {
      strBuffer.append(sTempOneLine);
    }

    System.out.println("result is : " + strBuffer.toString());
    System.out.println("success to read.");

  } finally {
    // make sure the streams are closed finally.
    close(reader);
    close(in);
  }
}
```

涉及API：

API名称	API描述
FileSystem.open	打开**HDFS**上指定的文件。

开发工程示例-文件写入/追加

数据写入流程：

1. 初始化**FSDataOutputstream**对象。
2. 使用**FSDataOutputStream**对象初始化**BufferedOutputStream** 。
3. 使用**BufferedOutputStream.write**写入**HDFS**数据。
4. 使用**BufferedOutputStream.flush**和**FSDataOutputStreadm.hflush()**将数据刷新到**HDFS**。
5. 关闭数据流。

```
private void setWriteResource() throws IOException {
    Path filepath = new Path(fileFullName);
    hdfsOutStream = fSystem.create(filepath);
    bufferOutStream = new BufferedOutputStream(hdfsOutStream);
}
```

```
private void setAppendResource() throws IOException {
    Path filepath = new Path(fileFullName);
    hdfsOutStream = fSystem.append(filepath);
    bufferOutStream = new BufferedOutputStream(hdfsOutStream);
}
```

```
private void outputToHDFS(InputStream inputStream) throws IOException {
    final int countForOneRead = 10240; // 10240 Bytes each time
    final byte buff[] = new byte[countForOneRead];
    int count;

    while ((count = inputStream.read(buff, 0, countForOneRead)) > 0) {
        bufferOutStream.write(buff, 0, count);
    }

    bufferOutStream.flush();
    hdfsOutStream.hflush();
}
```

涉及API：

API名称	API描述
FileSystem.create	创建指定文件的输入流，进行文件写入。
FileSystem.append	打开已存在的指定文件的输入流，进行文件写入。

目录

应用开发之SHELL篇

1. 登录**FusionInsight Manager**下载客户端安装包。

2. 执行**install.sh**安装客户端。

```
189-120-84-212:/tmp/FusionInsight-Client/FusionInsight_V100R002C60U10_Services_ClientConfig # ll
total 10140
drwx------ 7 root root     4096 Aug 23 20:44 HDFS
drwx------ 2 root root     4096 Aug 23 20:44 JDK
drwx------ 3 root root     4096 Aug 23 20:44 KrbClient
-rwx------ 1 root root     1811 Aug 23 20:43 README
drwx------ 7 root root     4096 Aug 23 20:44 Yarn
drwx------ 3 root root     4096 Aug 23 20:44 ZooKeeper
-rwx------ 1 root root      135 Aug 23 20:43 application.properties
-rwx------ 1 root root      781 Aug 23 20:43 bigdata_env.sample
-rw------- 1 root root     1322 Aug 23 20:43 ca.crt
-rwx------ 1 root root     4508 Aug 23 20:43 conf.py
-rw------- 1 root root      115 Aug 23 20:43 hosts
-rwx------ 1 root root      323 Aug 23 20:43 install.bat
-rwx------ 1 root root    13914 Aug 23 20:43 install.sh
-rwx------ 1 root root 10281273 Aug 23 20:43 jythonLib.jar
-rwx------ 1 root root      473 Aug 23 20:43 log4j.properties
-rwx------ 1 root root     2202 Aug 23 20:43 refreshConfig.sh
-rwx------ 1 root root     1736 Aug 23 20:43 switchuser.py
189-120-84-212:/tmp/FusionInsight-Client/FusionInsight_V100R002C60U10_Services_ClientConfig # ./install.sh /opt/client/
```

应用开发之SHELL篇

3.首先**source bigdata_env** 导入客户端的环境变量，从管理员处获取用户的密码使用**kinit**命令进行安全认证。

```
189-120-84-212:/opt/client # ll
total 10092
drwxr-xr-x 5 root root     4096 Aug 23 20:45 HDFS
drwxr-xr-x 3 root root     4096 Aug 23 20:45 JDK
drwxr-xr-x 4 root root     4096 Aug 23 20:45 KrbClient
drwxr-xr-x 3 root root     4096 Aug 23 20:45 Yarn
drwxr-xr-x 4 root root     4096 Aug 23 20:45 ZooKeeper
-rwxr-xr-x 1 root root      828 Aug 23 20:45 bigdata_env
-rwxr-xr-x 1 root root 10281273 Aug 23 20:45 jythonLib.jar
-rwxr-xr-x 1 root root     2202 Aug 23 20:45 refreshConfig.sh
-rwxr-xr-x 1 root root     1736 Aug 23 20:45 switchuser.py
189-120-84-212:/opt/client # source bigdata_env
189-120-84-212:/opt/client # kinit admin
Password for admin@HADOOP.COM:
```

```
189-120-84-212:/opt/client # hdfs dfs
Usage: hadoop fs [generic options]
    [-appendToFile <localsrc> ... <dst>]
    [-cat [-ignoreCrc] <src> ...]
    [-checksum <src> ...]
    [-chgrp [-R] GROUP PATH...]
    [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
    [-chown [-R] [OWNER][:[GROUP]] PATH...]
    [-copyFromLocal [-f] [-p] [-l] <localsrc> ... <dst>]
    [-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
    [-count [-q] [-h] [-v] [-t [<storage type>]] <path> ...]
    [-cp [-f] [-p | -p[topax]] <src> ... <dst>]
    [-createSnapshot <snapshotDir> [<snapshotName>]]
    [-deleteSnapshot <snapshotDir> <snapshotName>]
    [-df [-h] [<path> ...]]
    [-du [-s] [-h] <path> ...]
    [-expunge]
    [-find <path> ... <expression> ...]
    [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
    [-getfacl [-R] <path>]
    [-getfattr [-R] {-n name | -d} [-e en] <path>]
    [-getmerge [-nl] <src> <localdst>]
    [-help [cmd ...]]
    [-ls [-d] [-h] [-R] [<path> ...]]
    [-mkdir [-p] <path> ...]
    [-moveFromLocal <localsrc> ... <dst>]
    [-moveToLocal <src> <localdst>]
    [-mv <src> ... <dst>]
    [-put [-f] [-p] [-l] <localsrc> ... <dst>]
```

4.使用**HDFS shell**接口**hdfs dfs**编程。

应用开发之SHELL篇

```
189-120-84-167:/opt/client # hdfs dfs -ls /
Found 8 items
drwxrwxrwx   - hdfs   hadoop          0 2016-08-02 10:40 /app-logs
drwxr-xr-x   - hdfs   hadoop          0 2016-08-02 10:40 /datasets
drwxr-xr-x   - hdfs   hadoop          0 2016-08-02 10:40 /datastore
drwxr-x---   - flume  hadoop          0 2016-08-02 10:40 /flume
drwxrwxrwx+  - mapred hadoop          0 2016-08-02 10:40 /mr-history
drwxrwxrwx   - spark  hadoop          0 2016-08-02 10:40 /sparkJobHistory
drwxrwxrwx+  - hdfs   hadoop          0 2016-08-02 10:41 /tmp
drwxrwxrwx+  - hdfs   hadoop          0 2016-08-02 10:40 /user
189-120-84-167:/opt/client # hdfs dfs -mkdir /test
189-120-84-167:/opt/client # hdfs dfs -ls /
Found 9 items
drwxrwxrwx   - hdfs   hadoop          0 2016-08-02 10:40 /app-logs
drwxr-xr-x   - hdfs   hadoop          0 2016-08-02 10:40 /datasets
drwxr-xr-x   - hdfs   hadoop          0 2016-08-02 10:40 /datastore
drwxr-x---   - flume  hadoop          0 2016-08-02 10:40 /flume
drwxrwxrwx+  - mapred hadoop          0 2016-08-02 10:40 /mr-history
drwxrwxrwx   - spark  hadoop          0 2016-08-02 10:40 /sparkJobHistory
drwxr-xr-x   - admin  supergroup      0 2016-08-02 11:42 /test
drwxrwxrwx+  - hdfs   hadoop          0 2016-08-02 10:41 /tmp
drwxrwxrwx+  - hdfs   hadoop          0 2016-08-02 10:40 /user
```

创建并查看目录

```
189-120-84-167:/opt/client # hdfs dfs -ls /
Found 9 items
drwxrwxrwx   - hdfs   hadoop          0 2016-08-02 10:40 /app-logs
drwxr-xr-x   - hdfs   hadoop          0 2016-08-02 10:40 /datasets
drwxr-xr-x   - hdfs   hadoop          0 2016-08-02 10:40 /datastore
drwxr-x---   - flume  hadoop          0 2016-08-02 10:40 /flume
drwxrwxrwx+  - mapred hadoop          0 2016-08-02 10:40 /mr-history
drwxrwxrwx   - spark  hadoop          0 2016-08-02 10:40 /sparkJobHistory
drwxr-xr-x   - admin  supergroup      0 2016-08-02 11:45 /test
drwxrwxrwx+  - hdfs   hadoop          0 2016-08-02 10:41 /tmp
drwxrwxrwx+  - hdfs   hadoop          0 2016-08-02 11:43 /user
189-120-84-167:/opt/client # hdfs dfs -rm -r -skipTrash /test
Deleted /test
189-120-84-167:/opt/client # hdfs dfs -ls /
Found 8 items
drwxrwxrwx   - hdfs   hadoop          0 2016-08-02 10:40 /app-logs
drwxr-xr-x   - hdfs   hadoop          0 2016-08-02 10:40 /datasets
drwxr-xr-x   - hdfs   hadoop          0 2016-08-02 10:40 /datastore
drwxr-x---   - flume  hadoop          0 2016-08-02 10:40 /flume
drwxrwxrwx+  - mapred hadoop          0 2016-08-02 10:40 /mr-history
drwxrwxrwx   - spark  hadoop          0 2016-08-02 10:40 /sparkJobHistory
drwxrwxrwx+  - hdfs   hadoop          0 2016-08-02 10:41 /tmp
drwxrwxrwx+  - hdfs   hadoop          0 2016-08-02 11:43 /user
```

删除目录

应用开发之SHELL篇

下载文件

```
189-39-173-38:/opt/client # hdfs dfs -get /test/conf.py
189-39-173-38:/opt/client # ll
total 10120
drwxr-xr-x 5 root root     4096 Aug  2 11:42 HDFS
drwxr-xr-x 3 root root     4096 Aug  2 11:43 JDK
drwxr-xr-x 3 root root     4096 Aug  2 11:43 Kafka
drwxr-xr-x 4 root root     4096 Aug  2 11:43 KrbClient
drwxr-xr-x 3 root root     4096 Aug  2 11:43 SmallFS
drwxr-xr-x 4 root root     4096 Aug  2 11:43 Streaming
drwxr-xr-x 3 root root     4096 Aug  2 11:43 Yarn
drwxr-xr-x 4 root root     4096 Aug  2 11:43 ZooKeeper
-rwxr-xr-x 1 root root      832 Aug  2 11:43 bigdata_env
-rw-r--r-- 1 root root     4508 Aug  2 18:52 conf.py
-rwxr-xr-x 1 root root     4508 Aug  2 11:43 conf.py_bak
-rwxr-xr-x 1 root root 10281273 Aug  2 11:43 jythonLib.jar
-rwxr-xr-x 1 root root     2202 Aug  2 11:43 refreshConfig.sh
-rwxr-xr-x 1 root root     1736 Aug  2 11:43 switchuser.py
```

```
189-39-173-38:/opt/client # ll
total 10112
drwxr-xr-x 5 root root     4096 Aug  2 11:42 HDFS
drwxr-xr-x 3 root root     4096 Aug  2 11:43 JDK
drwxr-xr-x 3 root root     4096 Aug  2 11:43 Kafka
drwxr-xr-x 4 root root     4096 Aug  2 11:43 KrbClient
drwxr-xr-x 3 root root     4096 Aug  2 11:43 SmallFS
drwxr-xr-x 4 root root     4096 Aug  2 11:43 Streaming
drwxr-xr-x 3 root root     4096 Aug  2 11:43 Yarn
drwxr-xr-x 4 root root     4096 Aug  2 11:43 ZooKeeper
-rwxr-xr-x 1 root root      832 Aug  2 11:43 bigdata_env
-rwxr-xr-x 1 root root     4508 Aug  2 11:43 conf.py
-rwxr-xr-x 1 root root 10281273 Aug  2 11:43 jythonLib.jar
-rwxr-xr-x 1 root root     2202 Aug  2 11:43 refreshConfig.sh
-rwxr-xr-x 1 root root     1736 Aug  2 11:43 switchuser.py
189-39-173-38:/opt/client # hdfs dfs -put conf.py /test
189-39-173-38:/opt/client # hdfs dfs -ls /test
Found 1 items
-rw-r--r--   3 admin supergroup       4508 2016-08-02 18:48 /test/conf.py
```

上传文件

应用开发之SHELL篇

显示文件内容

```
189-39-173-38:/opt/client # hdfs dfs -cat /test/conf.py
1. Welcome to HDFS conf!
2. Welcome to HDFS conf!
3. Welcome to HDFS conf!
4. Welcome to HDFS conf!
5. Welcome to HDFS conf!

189-39-173-38:/opt/client # ll
```

```
189-39-173-38:/opt/client # cat appendfile
append to this file.
189-39-173-38:/opt/client # hdfs dfs -appendToFile appendfile /test/conf.py
189-39-173-38:/opt/client # hdfs dfs -cat /test/conf.py
1. Welcome to HDFS conf!
2. Welcome to HDFS conf!
3. Welcome to HDFS conf!
4. Welcome to HDFS conf!
5. Welcome to HDFS conf!

append to this file.
189-39-173-38:/opt/client #
```

将本地文件内容追加到**HDFS**上的文件

常用SHELL命令

命令类别	命令	命令说明
dfs	-cat	显示文件内容
	-ls	显示目录列表
	-rm	删除文件
	-put	上传目录/文件到HDFS
	-get	从HDFS下载目录/文件到本地
	-mkdir	创建目录
	-chmod/-chown	改变文件属组
	…	…
dfsadmin	-safemode	安全模式操作
	-report	报告服务状态
balancer	-threshold	容量均衡阈值

目录

应用开发规范

- **规范1**

  JAVA开发时，申请资源需及时释放，如**FSDataInputStream**、**FSDataOutputStream**、**BufferedOutputStream**、**BufferedReader**对象等。

- **规范2**

  HDFS不适用于存储大量小文件，大量小文件的元数据会占用**NameNode**大量内存。

- **规范3**

  HDFS中数据的备份数量**3**份即可，增加备份数量不能提升系统效率，只会提升系统数据的安全系数；当某个节点损坏时，该节点上的数据会被均衡到其他节点上。

- **规范4**

  如果有多线程进行**login**的操作，当应用程序第一次登录成功后，所有线程再次登录时应该使用**relogin**的方式。

本章总结

- 介绍了**HDFS**应用场景，并阐述和分析**HDFS**数据读取和数据写入的原理。
- 重点从**HDFS JAVA**开发和**SHELL**开发两方面讲解了**HDFS**应用的读写开发流程与常用接口。
- 介绍了在**HDFS**应用中的开发规范。

习题

1. 关于**HDFS**的文件写入，正确的是（ ）

A. 支持多用户对同一文件的写操作

B. 用户可以在文件任意位置进行修改

C. 默认将文件块复制成三份存放

D. 复制的文件块默认都存在同一机架上

2. Client在**HDFS**上进行文件写入时，**NameNode**根据文件大小和配置情况，返回部分**datanode**信息，（ ）负责将文件划分为多个**Block**，根据**DataNode**的地址信息，按顺序将块写入到每一个**DataNode**。

A．Client　　**B．Active NameNode**

C．DataNode　　**D．Standby NameNode**

习题

3. hdfs dfs命令中的-get和-put命令操作对象是（ ）

A. 文件

B. 目录

C. 两者都是

思考题

1. HDFS文件系统上，删除后进入回收站的数据能否恢复？如果能，使用什么客户端命令可恢复?

Thank you

www.huawei.com