前面学习了用nginx来提供HTTP的视频播放服务，其中我们说过关于编码问题，且也提到了ffmpeg，本篇文章就基于ffmpeg+ngnix+java+hls来实现转码、切片和视频播放方案。

一、ffmpeg的下载与安装(Windows版本)

官网地址为：`http://ffmpeg.org/`，其中下载地址为：`http://ffmpeg.org/download.html`，我们现在`windows`的本机上安装下看看，因此可以选中以下页面：

接下来会来到`https://www.gyan.dev/ffmpeg/builds/`页面，找到这里即可真正下载。

一开始解释了版本的意思：

```
git full - built from master branch with a large set of libraries
git essentials - built from master branch with commonly-used
libraries
release full - built from latest release branch with a large set of
libraries
release essentials - built from latest release branch with
commonly-used libraries
```

那我们就下载这个`git full`版本吧。

release

Links

https://www.gyan.dev/ffmpeg/builds/ffmpeg-release-full.7z
SHA256: d3b2eab226c54eb68a1777d92a8df79a5ee8abf39d0efd9f852e7be144f020cc

https://www.gyan.dev/ffmpeg/builds/ffmpeg-release-essentials.7z (22MB)
SHA256: 1f85f6340af5b15ef3ba335c9a35810b16c8b58fca1cb0d1542cc40a5be4c647

https://www.gyan.dev/ffmpeg/builds/ffmpeg-release-essentials.zip (74MB)
SHA256: d80ad9b9dc4e90c07190737f759bfe24f16fdf6e1095d9c959d9c5f592ba7081

https://www.gyan.dev/ffmpeg/builds/ffmpeg-release-full-shared.7z
SHA256: bed0a5857cf6578a62943aa248796e1aaad059e510a55bfeaef2e83623886716

Mirror

https://www.gyan.dev/ffmpeg/builds/ffmpeg-release-github

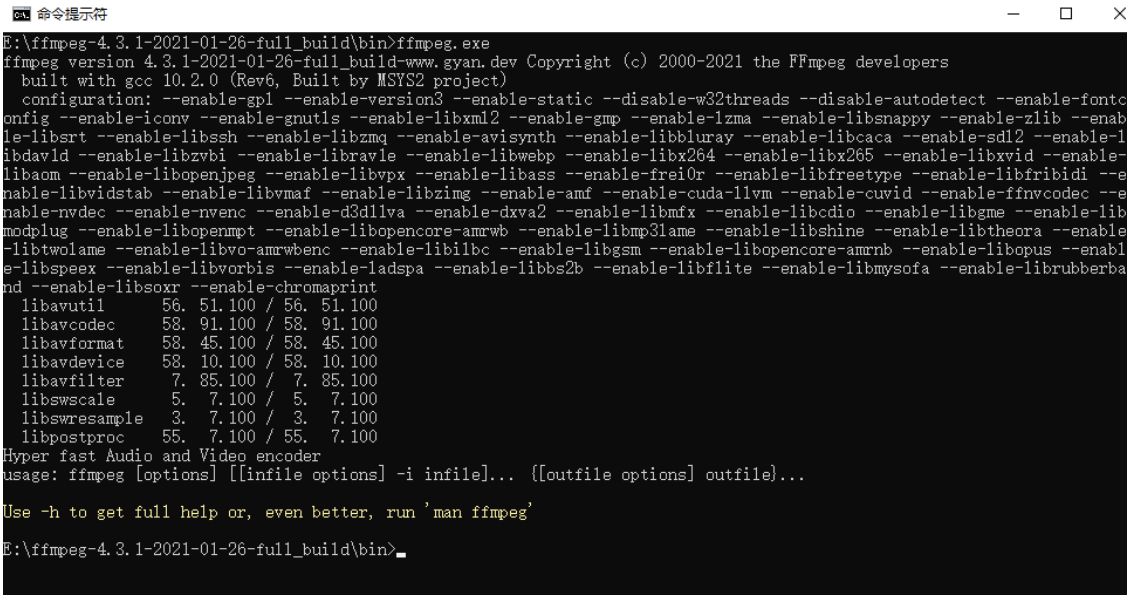
Version

4.3.1-2021-01-26

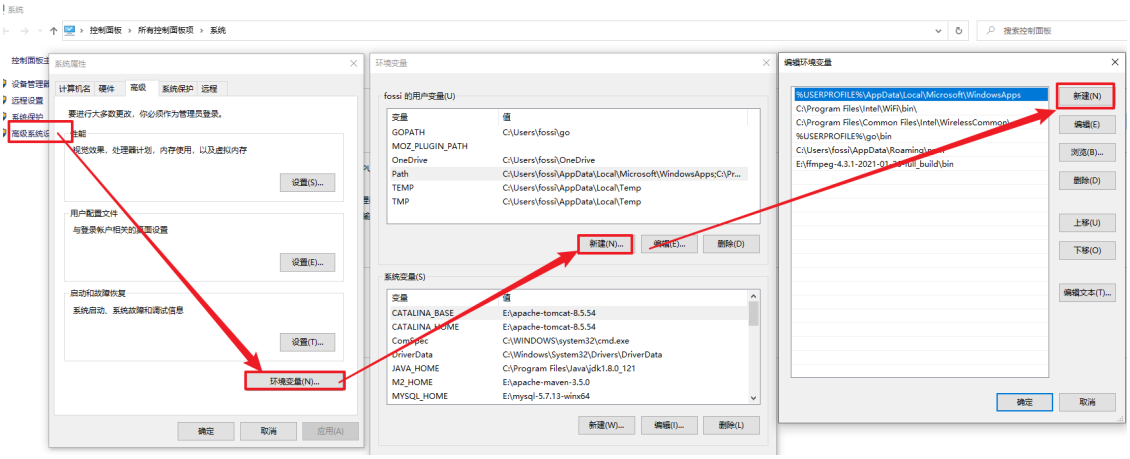
Source code

https://github.com/FFmpeg/FFmpeg/commit/666d2fc6e2

经过漫长的等待之后，终于下载完毕。解压进入 `bin` 目录，这里显然是可执行文件的目录了，我们来试一把：`ffmpeg.exe`



看来没啥问题，我们把它配置成环境变量吧，不需要每次都到这个目录下来执行。这个就非常简单了，我们只需要将 `bin` 目录拷贝下来，打开环境变量添加到 `path` 中即可：



我们随意打开命令行输入 `ffmpeg` 看下效果：

```
Windows PowerShell
PS C:\Users\fossi\Desktop> ffmpeg
ffmpeg version 4.3.1-2021-01-26-full_build-www.gyan.dev Copyright (c) 2000-2021 the FFmpeg developers
  built with gcc 10.2.0 (Rev6, Built by MSYS2 project)
  configuration: --enable-gpl --enable-version3 --enable-static --disable-w32threads --disable-autodetect --enable-fontc
onfig --enable-iconv --enable-gnutls --enable-libxml2 --enable-gmp --enable-lzma --enable-libsnappy --enable-zlib --enab
le-libsrt --enable-libssh --enable-libzmq --enable-avisynth --enable-libbluray --enable-libcaca --enable-sdl2 --enable-l
ibdav1d --enable-libzvbi --enable-librav1e --enable-libwebp --enable-libx264 --enable-libx265 --enable-libxvid --enable-
libaom --enable-libopenjpeg --enable-libvpx --enable-libass --enable-frei0r --enable-libfreetype --enable-libfribidi --e
nable-libvidstab --enable-libvmaf --enable-libzimg --enable-amf --enable-cuda-llvm --enable-cuvid --enable-ffnvcodec --e
nable-nvdec --enable-nvenc --enable-d3d11va --enable-dxva2 --enable-libmfx --enable-libcdio --enable-libgme --enable-lib
modplug --enable-libopenmpt --enable-libopencore-amrwb --enable-libmp3lame --enable-libshine --enable-libtheora --enable
-libtwolame --enable-libvo-amrwbenc --enable-libilbc --enable-libgsm --enable-libopencore-amrnb --enable-libopus --enabl
e-libspeex --enable-libvorbis --enable-ladspa --enable-libbs2b --enable-libflite --enable-libmysofa --enable-librubberba
nd --enable-libsoxr --enable-chromaprint
  libavutil      56. 51.100 / 56. 51.100
  libavcodec     58. 91.100 / 58. 91.100
  libavformat    58. 45.100 / 58. 45.100
  libavdevice    58. 10.100 / 58. 10.100
  libavfilter     7. 85.100 /  7. 85.100
  libswscale      5.  7.100 /  5.  7.100
  libswresample   3.  7.100 /  3.  7.100
  libpostproc    55.  7.100 / 55.  7.100
Hyper fast Audio and Video encoder
usage: ffmpeg [options] [[infile options] -i infile]... {[outfile options] outfile}...

Use -h to get full help or, even better, run 'man ffmpeg'
PS C:\Users\fossi\Desktop>
```

好了，`ffmpeg` 的下载和安装就轻松搞定了。下面我们得手动试一下 `ffmpeg` 的切片转码功能。

二、手动执行ffmpeg命令

新建文件夹，准备好源视频。我的视频是一个大TS文件，比如我本地的是 `C:\Users\fossi\Desktop\切片测试\happyedu.ts` ，这是一个大约1分32秒的视频。

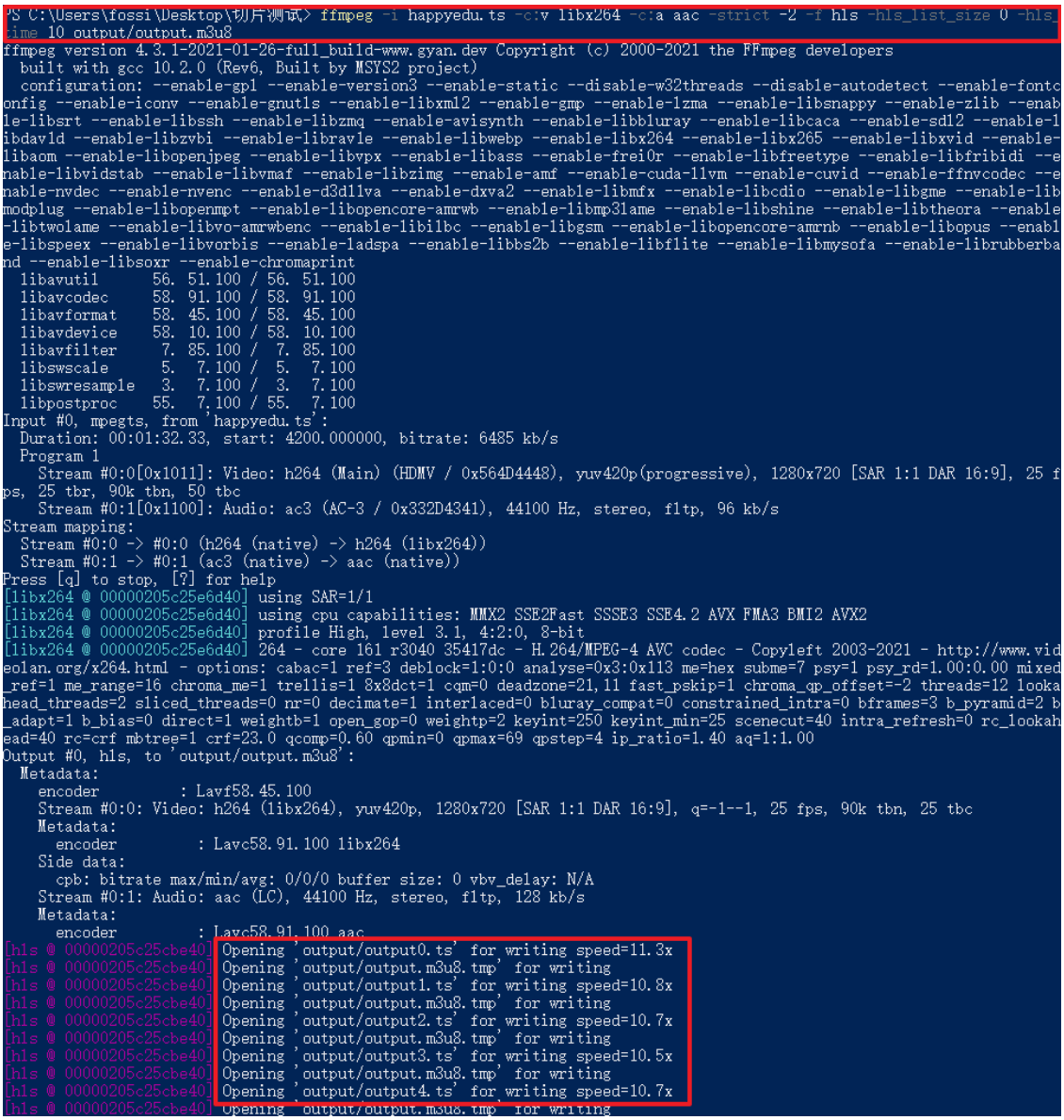
下面我准备切片，经过之前的学习，我们知道H5对于视频播放的编码是有一定要求的，因此我就按照最保险的方式，将其转码为H264的视频编码和AAC的音频编码。如何来做呢？ffmpeg极其强大，我用下面这一条命令实现了转码和切片的目的。

> 关于ffmpeg命令的说明，阮一峰老师的文章推荐给大家，我这里就不重复说明了：http://www.ruanyifeng.com/blog/2020/01/ffmpeg.html

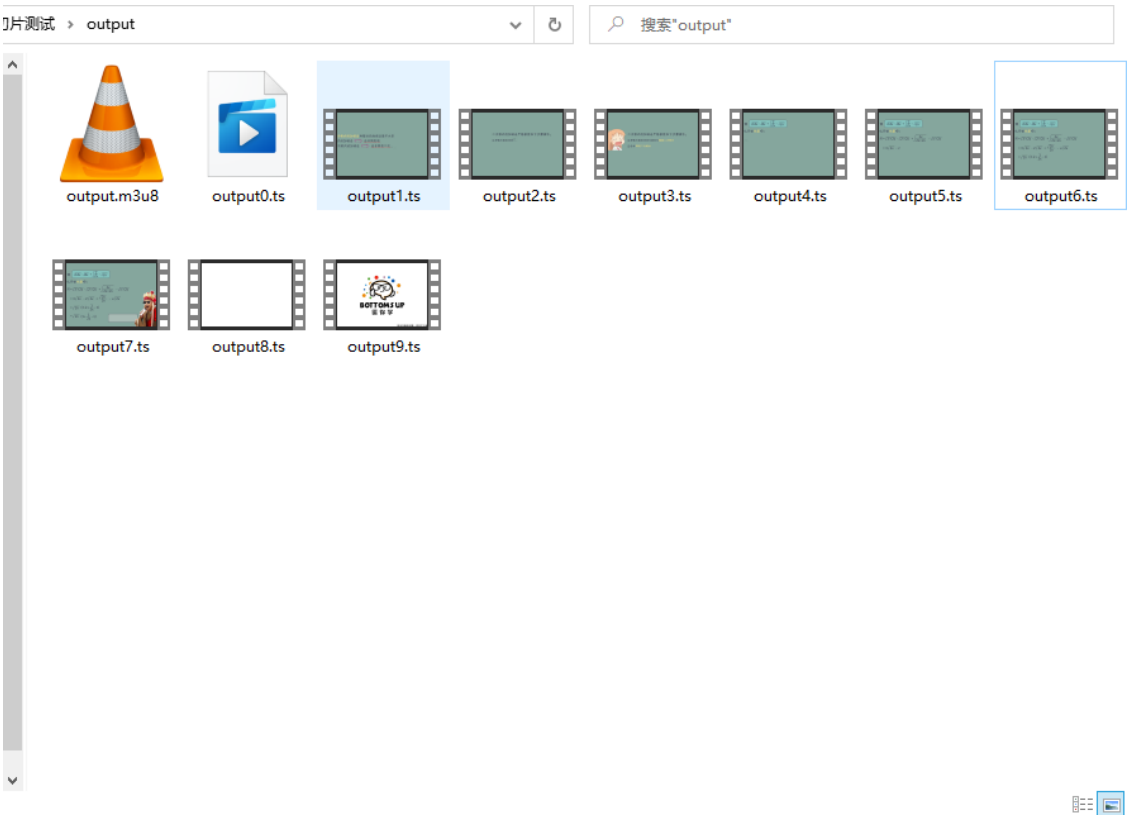
```
ffmpeg -i happyedu.ts -c:v libx264 -c:a aac -strict -2 -f hls -hls_list_size 0 -hls_time 10 output/output.m3u8
```

- `ffmpeg -i happyedu.ts`：表示我们输入的视频是 `happyedu.ts`
- `-c:v libx264`：将视频转为H.264编码，一般用编码器 `libx264`
- `-c:a aac`：将音频转为 `aac` 编码
- `-hls_time n`：设置每片的长度，默认值为2。单位为秒
- `-hls_list_size n`:设置播放列表保存的最多条目，设置为0会保存有所片信息，默认值为5
- `-strict -2` 这个意思没找着，有清楚的读者朋友知道的提醒下?

下面就是执行此命令，注意，我需要提前创建好 `output` 文件夹：



正确执行后， `output` 文件夹下就出现了：



我随便打开了一个切片，看了下视频编码和音频编码都是符合条件的。

将切出来的output文件夹放到 `nginx` 的映射目录下，打开 `vlc` 播放器进行播放，我的播放链接为 `http://localhost/output/output.m3u8`

或许有同学好奇，HLS协议到底啥时候开始下载切片的？即如何提高用户体验的？我们可以用 `wireshark` 抓个网络包来试试。

可以看到，HLS协议的运作流程十分简明，第一步是下载 `m3u8` 索引文件，第二步是按次序下载TS文件。在我们播放第一个TS文件的时候，播放器已经在默默帮我们下载第二个、第三个，这样就像一条河一样源源不断地输送过来，用户观看视频自然也不会卡。这个听起来是不是很像直播流，没错，直播也是可以依靠HLS这个协议工作，核心上就是维护一个不断更新的m3u8文件从而实时拉流，播放器是主动请求的一方，所以说，这也离不开播放器良好的逻辑设计。

三、JAVA代码调用ffmpeg命令

好了，手动试起来感觉ffmpeg是真的很爽，不过上面终究只是手动在试验，我们需要借助代码来自动转码切片才行，这才是本篇文章的终极目的。那就开始打开我们的老基友IDEA吧！

代码参考了`https://github.com/eguid/FFCH4J`项目。可以先下载下来源代码，然后找到`cc.eguid.commandManager.test.Test`，可以看到里面有一个`main`方法和若干测试方法，我是直接修改的`test4()`方法，将其具体的指令改为我自己的指令，发现是可以正常跑出来的。不过为了能为我所用，我将其改造为了`springboot`项目，希望对要用的朋友有点用。代码我托管到了`github`，项目地址为`https://github.com/sunweiguo/ffmpegtest`.

首先我的代码入口是一个`controller`，后面试验通过页面刷新去触发代码生效。

```
@RestController
public class TestController {

    @Autowired
    private CommandManager commandManager;

    @RequestMapping("execute")
    public String execute() throws InterruptedException {
        System.out.println("开始转码切片");
        String id = commandManager.start("ID任务00000001",
"E:\\ffmpeg-4.3.1-2021-01-26-full_build\\bin\\ffmpeg -i
C:\\Users\\fossi\\Desktop\\切片测试\\happyedu.ts -c:v libx264 -c:a
aac -strict -2 -f hls -hls_list_size 0 -hls_time 10
C:\\Users\\fossi\\Desktop\\切片测试\\output\\output.m3u8");
        Thread.sleep(30000);
        System.out.println("转码切片结束"+id);
        //停止任务
        commandManager.stop(id);
        return "success";
    }
}
```

可以看到，核心是`commandManager.start()`方法。这个方法里面，创建了执行指令的进程和输出打印信息的线程。并且将当前的任务放到了一个`ConcurrentHashMap`中，便于管理任务。具体的`start()`方法为：

```
@Override
public String start(String id, String path) {
  if (id != null && path != null) {
```

```
    CommandTasker tasker = taskHandler.process(id, path);
    System.out.println(tasker.toString());
    if (tasker != null) {
      //返回true，说明任务添加到map成功，且不存在重复任务正在执行
      boolean ret = taskDao.add(tasker);
      if (ret) {
        return tasker.getId();
      } else {
        // 持久化信息失败，停止处理
        taskHandler.stop(tasker.getProcess(), tasker.getThread());
        System.err.println("持久化失败，停止任务！");
      }
    }
  }
  return null;
}
```

一直进去，就会找到这个方法：

```
public static CommandTasker createTasker(String id, String command,
OutHandlerMethod ohm) throws IOException {
  // 执行本地命令获取任务主进程
  Process process=exec(command);
  // 创建输出线程
  OutHandler outHandler=OutHandler.create(process.getErrorStream(),
id,ohm);

  CommandTasker tasker = new CommandTasker(id,command, process,
outHandler);

  return tasker;
}
```

可以看到，创建进行和输出线程就是在这里做的。创建进程去执行指令我们可以理解，但是输出线程有什么用呢？我们可以根据打印的信息去判断执行指令是否失败，以及是否已经执行完毕。在 `com.swg.demo.handler.OutHandler` 中：

```
/**
 * 执行输出线程
 */
@Override
public void run() {
  String msg = null;
  try {
    System.out.println(id + "开始转码切片！");
```

```
    while (desstatus && (msg = br.readLine()) != null) {
      ohm.parse(id,msg);
      if(ohm.isbroken()) {
        System.err.println("检测到到错误，本次任务终止，开始记录数据库本次操作状态为切片失败");
        //TODO
        // 1、如果发生异常中断，设置数据库状态为失败
        // 2、commandManager.stop(id);删除本次任务，停止线程和执行进程
      }
    }
    Thread.sleep(100);
    //走到这里，没有发生异常，没有任何输出日志了，这个时候说明已经完成了，这个时候，就认为是处理成功了
    System.out.println("切片成功！！！！！！！");
    //TODO
    // 1、更新数据库状态为切片成功，这样前端就可以知道哪些任务是切片中、切片成功、切片失败了
    // 2、程序定时任务去扫描状态为切片成功或切片失败，销毁进程和线程
  } catch (IOException e) {
    System.out.println("发生内部异常错误，自动关闭[" + this.getId() + "]线程");
    destroy();
  } catch (InterruptedException e) {
    e.printStackTrace();
    destroy();
  } finally {
    if (this.isAlive()) {
      destroy();
    }
  }
}
```

好了，代码逻辑还是要自己去看，不过声明下，上面所述是核心代码，仅供参考而已，如果你恰好业务需求跟我一样，那么看懂这个流程再做优化和修改应该都是没有问题的。下面我启动项目，访问`http://localhost:8080/execute`，这个时候，控制台就会输出转码切片的信息，且指定目录已经生成了对应的切片文件。

四、配合nginx实现在线视频播放

最后，我设想了如果要做一个在线视频审核系统的话，实时切片是不现实的，需要有一个切片任务的控制，切片完成后管理后台拼接实际播放地址，提供给审核人员进行视频审核。下面是我设想的一个流程：

这个方案本身优缺点都说下吧：

优点不言而喻，如果是走HTTP形式的播放，那么这种小切片对于视频的播放流畅度还是很有帮助的。

缺点也同样明显，第一作为管理系统级别的视频播放方案我感觉还是稍微复杂了，且ffmpeg对CPU的消耗极大，上面的图我考虑到可能会影响管理后台本身的一些业务逻辑，因此考虑将其放到一个单独的切片服务器上去处理。并且我们的视频是存放在FTP上的，本身容量就告急，这里还需要多存一份。

其他的方案考虑。

1. 不切片，一个大TS文件，系统帮助他生成对应的索引文件即可，后面将实际试验下体验效果。
2. 直接用VLC播放FTP，效果很好且极其简单，不过缺点同样致命：暴露了FTP的IP、用户名和密码，哪怕只是提供只读权限，也觉得不合适。

其实写到这里，做一个管理系统级别的视频在线播放，其实不难，如果是面向广大用户，则需要用CDN服务，跟这个就不是一个量级了。转码切片的系统也将考虑很多因素，比如并发、稳定性、可用性等因素。以上只是个人的思考，项目用什么方案还需更多的实验和讨论。本篇文章就到这里吧~

PS ：代码仅供参考，请忽略细节。