神马 16 核的服务器你让我单线程跑 ffmpeg

周末了，也该把上周工作中用到的雕虫小技整理下，写成博客，也方便我以后查阅。

最近需要用 ffmpeg 进行视频转码，考虑到 ffmpeg 这个工具本身支持多线程，而服务器也是多核的，想最大程度发挥服务器的计算能力。显然不能用单线程了，但是线程并不是越多越好，需要找到个平衡点。这就需要用 time 命令进行计时，并将该屏幕的输出重定向到文件，以便后期进行分析。

ffmpeg 编码的介绍网上有很多，我这里也有些示例（http://www.cnblogs.com/MikeZhang/archive/2012/07/17/videoCodec.html 中的“三、视频编码工具 ”中有介绍）。今天主要介绍下 ffmpeg 的转码，以及 time 命令的重定向问题。

假设我这里有个 in.mp4 的视频文件，分辨率为 640*480，码率为 580kbps，需要降低分辨率为 320*240，降低码率为 290kbps（这个参数和分辨率同时使用时不一定起作用，特别是中途退出时）

普通降低分辨率，码率的命令：

```
ffmpeg -y -i in.mp4 -s 320x240 -b 290000 out290.mp4
```

带 thread 参数的命令：

```
ffmpeg -y -threads 2 -i in.mp4 -s 320x240 -b 290000 out290.mp4
```

这里用了两个线程。

用 time 统计时间的命令：

```
time ffmpeg -y -threads 2 -i in.mp4 -s 320x240 -b 290000 out290.mp4
```

输出重定向命令：

```
(time ffmpeg -y -threads 2 -i in.mp4 -s 320x240 -b 290000 out290.mp4) 2>1.txt
```

由于输出为标准出错，所以用 2>1.txt

也可使用如下命令：

```
(time ffmpeg -y -threads 2 -i in.mp4 -s 320x240 -b 290000 out290.mp4) >& 1.txt
```

为了同时在屏幕上输出，可以用 tee 命令。首先将标准出错重定向到标准输出，然后通过通道传给文件，命令如下：

```
(time ffmpeg -y -threads 2 -i in.mp4 -s 320x240  out290.mp4) 2>&1 | tee 1.txt
```

如果想测试下服务器到底支持多少个线程比较好，就需要写个脚本运行，我这里有一个，仅供参考：

```
#! /bin/bash

echo "Input num : "
read num
for (( i=1; i<=$num; i=i+1 ))
do
     #(time ffmpeg -y -threads $i -i in.mp4 -s 320x240
-vcodec libx264 -vpre fast out290.mp4) >& $i.txt
     (time ffmpeg -y -threads $i -i in.mp4 -s 320x240
-vcodec libx264 -vpre fast out290.mp4) 2>&1 | tee $i.txt
done
```

好吧，启动脚本，让服务器飞一段时间，到时用 tail 命令看结果就是了。