**一：模型抽象**

转码系统可以抽象为一个输入帧数据、消耗cpu与内存资源进行处理。输出写文件的系统。该系统的输入与资源都是不稳定的。

因此需要针对不稳定输入、不稳定资源俩方面来控制转码速率。一共16种可能的场景:

流输出

流输入

CPU

转码器

输入较少、cpu占用较低

输入增加、cpu占用较低

输入很多、cpu占用较低

输入抖动、cpu占用较低

输入较少、cpu占用增加

输入增加、cpu占用增加

输入很多、cpu占用增加

输入抖动、cpu占用增加

输入较少、cpu占用较高

输入增加、cpu占用较高

输入很多、cpu占用较高

输入抖动、cpu占用较高

输入较少、cpu占用抖动

输入增加、cpu占用抖动

输入很多、cpu占用抖动

输入抖动、cpu占用抖动

以下为分为俩方面的控制方法:

**二:输入控制:输入平缓**

算法:记录每帧读取间隔t，设置阀门值limit\_fps。初始limit\_fps=video\_in\_fps

每1000ms计算limit\_t=1000/limit\_fps，x>=y时, limit\_t++,x<y时limit\_t- -。重置x、y

当t<=limit\_t时。延长t到limit\_t，计数 x++

当t>limit\_t时，t不变，计数y++

**三:CPU拥塞控制**

一、基础算法：TCP拥塞控制（详见[TCP拥塞控制](https://zhuanlan.zhihu.com/p/59656144)）

cpu采集每1s判断一次

一共划分为20档 0\*fps 0.1\*fps 0.2\*fps ----- 2.0\*fps

最初为1档 0.1\*fps

保证每秒转码帧率不超过当前挡位设定

基础为TCP拥堵算法：cwnd初始为1档ssthresh为20档，当cpu大于70(该值可由外部设置)时认为拥堵。进行减半退避，cwnd重置1。

1. 当cwnd <  ssthresh,使用慢启动算法，

2. 当cwnd >= ssthresh,使用拥塞控制算法。

二、cpu算子调控

在如上基础上，每个轮次cpu获取的状态影响慢启动与拥塞控制算法的算子:

cpu位于0-20时，算子为2

cpu位于20-40时，算子为1.5

cpu>40时，只用拥塞控制算法线性增加，算子为1

三、熔断机制

在如上基础上，加入熔断机制。

当出现cpu>80的情况，速率降低为0。熔断持续5个轮次，之后再重新启动。重新启动后如果立即又进入熔断，熔断时间乘2增长。如过重启后cwnd增长到20，熔断时间重置

四、反馈机制

在如上基础上，加入消息检测。

检测写入一帧音频的时间记录为x。记录x\_avg=x\_avg\*0.9+x\*0.1。当x>1.5\*x\_avg时，认为拥堵，当x>2\*x\_avg时，熔断。

CPU超过70、拥堵、超时

CPU超过80、熔断