本设计所使用的系统版本：

(1).CentOS Linux release 7.6.1810 (Core)

(2).kernel version: 4.18.0-rc1+

(3).Mediasdk(1.27)git commit version:acabda62ffa39f6889316b02381463af3f8dcafd

首先我们先对各种算法的重要参数设置值做一下说明。LA-ICQ和LA的预测深度-lad设为25。CBR，VBR，LA基准码率-b设为2000kbps。CQP这一码率控制算法在Media SDK中的实现需要输入的控制参数有-qpi，-qpp，-qpb三个参数，分别代表给I帧、P帧、B帧分配固定的QP值。我们通过-qpi，-qpp，-qpb三个参数的设置把转码出来的视频平均码率控制在2000kbps。同样的，LA-ICQ方法也灵活控制基准curQP值来把转码出来的视频平均码率控制在2000kbps。视频的GOP长度统一设为25，分辨率设为1080P，转码异步深度设为4。

该算法中各变量的含义如下：

Width：输入视频的图像宽度。

Height：输入视频的图像高度。

IntraCost：这个参数描述的是这一帧应用帧内预测的代价，是该帧图像只使用帧内预测时所有宏块SAD或SATD的累加和。这个变量可以用于衡量一帧图像的复杂程度。

InterCost：这个参数描述的是这一帧帧间预测的代价，是该帧图像允许使用帧间预测时所有宏块SAD或SATD的累加和。该变量表示当前相对于其参考帧的残差图像的复杂程度。若当前帧为I帧，则不使用帧间预测，该变量等于当前帧帧内预测的代价。该变量可以衡量实际转码一帧图像的复杂程度。

EstimatedRate：这个变量的作用是估计码率，在预转码阶段，它会按照QP值从1到51对这一帧进行转码，根据图像中各宏块选择的跳过、帧内预测、帧间预测的模式，宏块的SAD或SATD，以及量化步长，估计输出的码率。然后把由这51个QP值计算出来的码率放在一个数组中，我们在正式转码时就可以利用我们得出的当前帧的QP值取出我们提前计算得到的预测码率值，然后就可以将这一数值与我们的门限码率进行比较了。

我们的算法需要预先设定的参数有：

目标质量Target Quallity：本算法对码率的控制主要是通过对当前帧QP值的控制。这个参数初始化了QP的值，而我们算法的控制是在这个初始化QP值的基础上进行的。这个参数可以控制视频整体质量的高低。

控制深度Rate Control Depth：有一些视频的瞬时码率波动比较大，如果我们用每一帧的瞬时码率来进行是否超过门限码率的判断，很可能会出现码率控制程度太大以至于严重影响视频画质，或者对码率的限制不足码率依旧很大的情况。为了解决这个问题，本算法对一定周期的视频帧的数据长度进行统计，根据周期内的总数据长度计算出该周期内的平均码率，然后与门限码率进行比较，这样就很好地解决了误差问题。这个参数指定码率统计周期的大小和Look Ahead的深度。因为视频在一个GOP中，帧的大小往往与帧类型相关，通常帧的数据长度满足I帧>P帧>B帧的关系，码率统计周期最好包含相同数量的I帧、P帧和B帧，使得码率统计更加准确，因此控制深度应选取为GOP大小的整数倍，本算法中选取的控制深度等于GOP大小，即25帧。

码率门限Max Bitrate：本算法与Intel Media SDK中LA-ICQ码率控制算法的不同之处就是设计了一个自动的码率门限，这个门限码率可以根据设置的视频分辨率自动调整。当码率统计结果超出该门限时，就对当前帧的QP值进行惩罚， 一定程度上降低当前帧图像质量，对码率进行限制。

代码主要修改的目录路径为Intel 官方给出的Media SDK开源代码中的Media/\_studio/mfx\_lib/encode\_hw/h264/src/mfx\_h264\_encode\_hw\_utils.cpp。

我们主要改进的部分为LookAheadCrfBrc部分，该部分主要分为5个小部分，Init初始化，GetQP获得当前帧QP值，GetqQpForRecode记录QP值，PreEnc预转码部分，Report打印结果。我们做出改进的地方主要在PreEnc，GetQP以及Report这三个部分。

原先代码的PreEnc预转码部分所做的工作为提前计算即将转码帧的intreCost，interCost，propCost信息并存储，方便在正式编码GetQP部分进行相关的计算。

而我们改进的算法在PreEnc预编码部分进行了一个预测长度内宏块级别的码率预测以及按照原先方法所得无预测QP值的提前计算。

首先是宏块级别的码率预测，我们的方法是计算从0-51各个QP值下的mb.dist数值，然后在乘以一些固定参数。



其中LaicqMultiplier，INTRA\_QSTEP\_COEFF为可设定的固定参数，QSTEP[qp]为与当前帧QP值相关的一个数值，可查表获得。其中包含码率预测值的部分为mb.dist。



之前提到过intraCost表示的是 一帧图像内容的复杂度，这里的bitCostLambda表示的就是可以去除的冗余度，包括与之前帧的相关性和帧内的冗余。

对此处求得的帧中各个宏块码率预测值求平均就得到了相应QP值下的码率预测值，单位为比特每宏块。

接着是按照原先方法所得无预测QP值的提前计算。其实计算方法和原先方法基本一致。差别在于两个地方，一是我们将无预测QP值从原代码中的GetQP部分改到在PerEnc部分完成。二是原先的代码只计算了当前帧的无预测QP值，在此处我们提前计算了一个预测长度的帧的无预测QP值，并存储起来供GetQP部分使用。

原先代码的GetQP部分所做的工作为进行无预测QP值的计算，并用计算出来的QP值用于后续的编码。

**我们在Report部分加入了一段代码，在编码完成一帧图像之后，将该帧的数据长度存储在一个vector容器中，并判断当数据个数大于12（GOP长度的一半）的时候，剔除最早进入vector容器的一个数据。然后统计vector容器中的总数据长度，得到过去已经编码完成的视频帧的总数据长度PastTotalEstRate。**