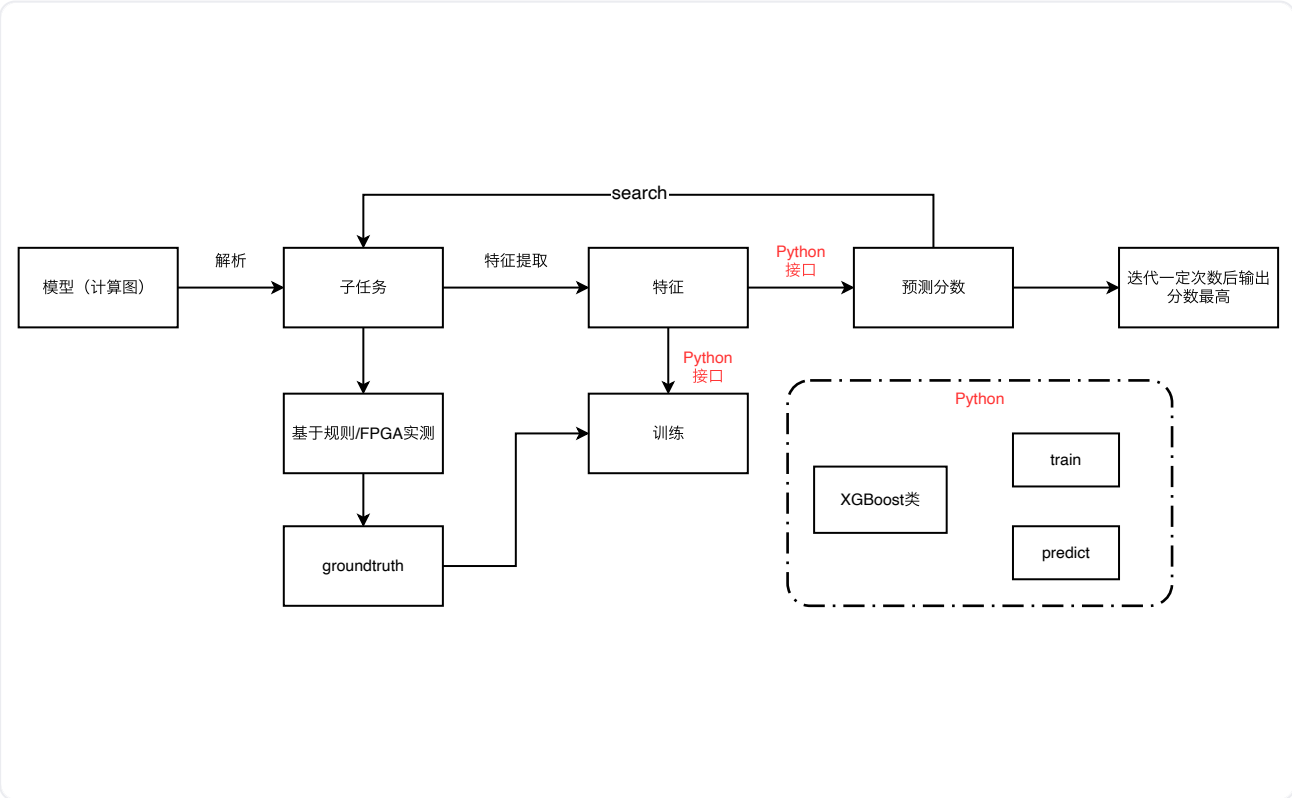
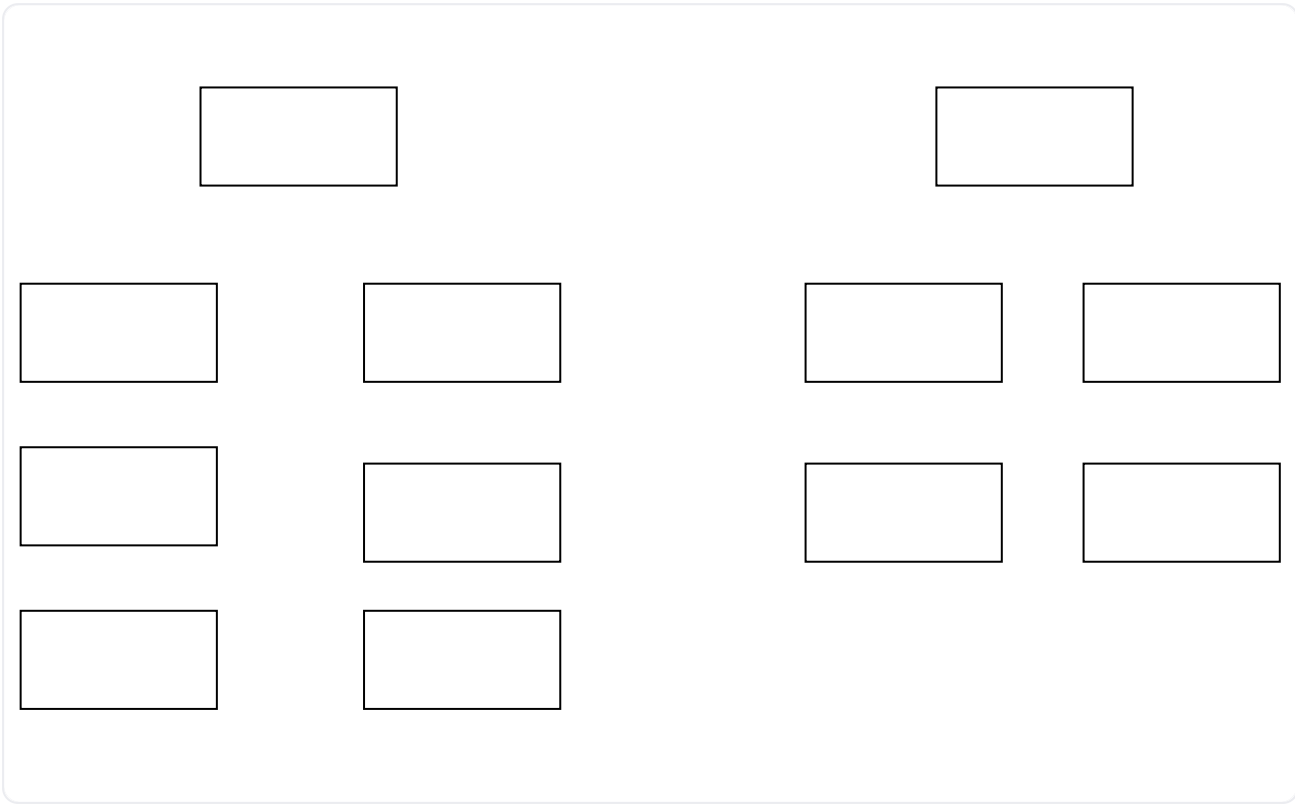
先假设特征都已经定义好了，实现了特征提取的相关方法，实现了反馈分数后search的功能。

参考autoschedule，设计如下系统：



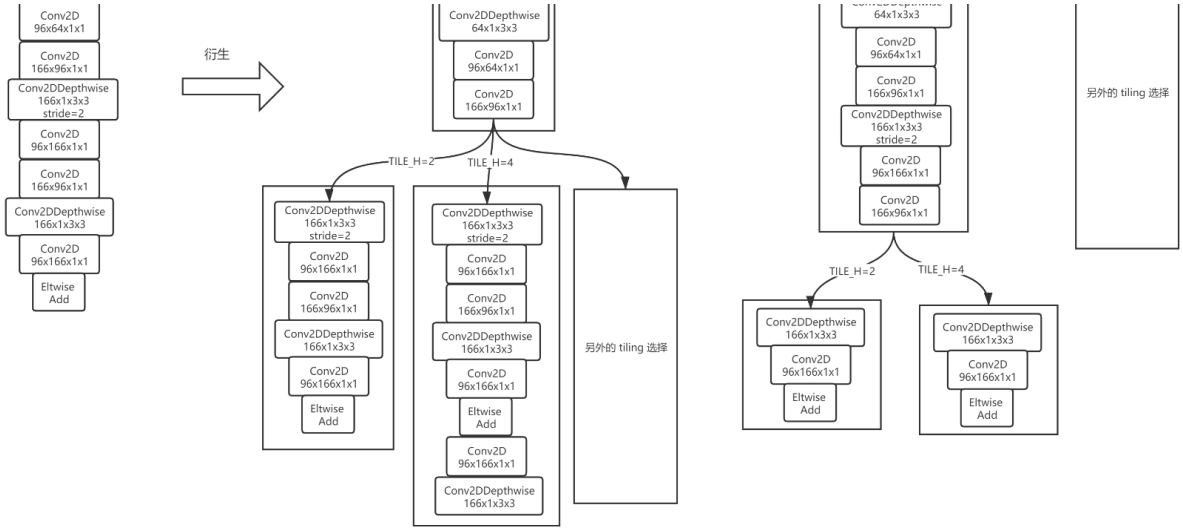
整体思路：
模型调优主体用C++实现，
首先读入基础模型，
解析模型并切分成多个子任务（直接对整个模型做特征提取工作量过大且不具备通用性），
预测模式：提取子任务特征，调用XGBoost类Python接口的predict函数，传入特征参数，返回预测分数。进化搜索子任务的新结构。

重要的一点：划分子任务的时候，我们想要搜索分tile时利用率最大，因此分tile之后的整个结构应该作为一个任务，然后对这个tile结构进行微调；理论上相当于把tile之后的几个子层融合。
每个层都进行分tile和一个总的分tile。



分tile是因为缓存空间不够，分批次才能实现。
如何分tile，分几个，参数是几，都可以search出来。

单纯对计算的优化，需要搜索所有的计算组合
主要是tile的组合，需要进行搜索，这么一看搜索的目的也很简单，就是要找到最优tile方式。



如果用穷举法，肯定是可以的，筛选出其中不符合硬件条件的case，对剩下的case进行评估获得最优tile方式。
就是把tile作为一个特征，并且相关联的特征包括DMA读取以及tile后面有几层；
但是其实不只是tile这一个特征，
卷积运算数学本身的优化目前暂时不考虑，也没有设计相关的特征

针对于tile来说，只要其中几个特征就行，先做tile相关的调优。

肯定要考虑的特征：

1. 卷积计算的卷积核，输入图片尺寸通道数等，代表着浮点乘法的数量即数学运算
2. 当前卷积核是否分tile，是，加到tile里面，
3. vector1的DMAread，vector2的DMAread有没有被隐藏
4. 进入vector2时

计算相关特征

缓存访问

算术强度

缓存分配

外围相关

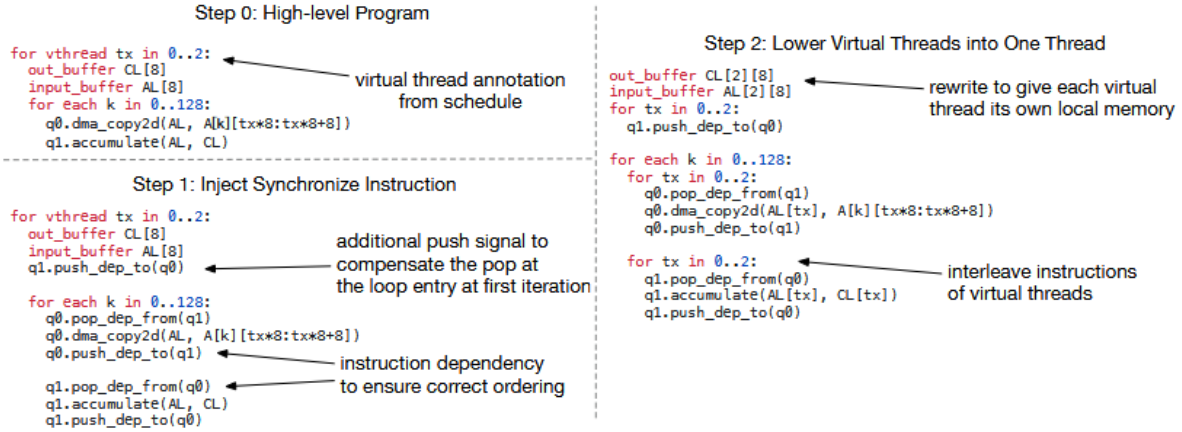
用c调用xgboost模型进行预测，

将模型的操作都记录字符并储存入vector数组中

隐藏时延（DMA读取操作）

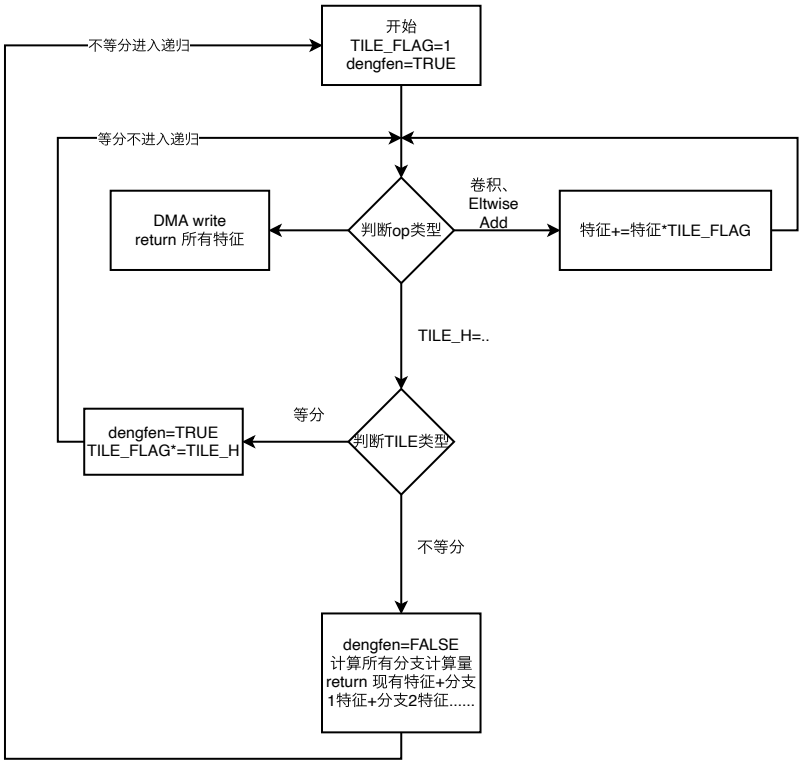
gives direct control to the programmer and compiler over how hardware tasks execute. By using low-level synchronization primitives exposed by the hardware, we can effectively hide latency.

通过使用硬件公开的底层同步原语，我们可以有效地隐藏延迟。



显式同步指令流（降低将带有虚拟线程的高级并行程序转换为显式同步程序模型的过程）

TVM使用虚拟线程来隐藏内存访问延迟。隐藏将数据加载到加速器中的一些延迟。


开始
TILE_FLAG=1
dengfen=TRUE
不等分进入递归
等分不进入递归
判断op类型
卷积、
Eltwise
Add
特征+=特征*TILE_FLAG
DMA write
return 所有特征
TILE_H=..
判断TILE类型
等分
dengfen=TRUE
TILE_FLAG*=TILE_H
不等分
dengfen=FALSE
计算所有分支计算量
return 现有特征+分支
1特征+分支2特征.....