本周任务

1. 读FTRANS
2. 确定Compression的Search Space
3. 计算稀疏矩阵新建索引之后带来的开销
4. 如何修改dense模式下的performance model以得到sparsity模式下的performance model（没有具体的解决）

任务一：确定Compression的Search Space

- **BBS’**

  把每行分成同样大小的banks，bank内部进行细粒度的剪枝，同一行的bank内剪枝率是相同的，不同行的bank剪枝率可以不相同。

		0.2	-0.6		0.4		0.6
0.4	-0.3	0.4		0.2	-0.4		0.5

0.2	0.1	0.2	-0.6	0.1	0.4	-0.1	0.6
0.4	-0.3	0.4	0.1	0.2	-0.4	0.1	0.5
0.7	-0.1	-0.3	0.1	0.5	-0.1	0.5	0.1
-0.1	0.6	-0.5	0.3	-0.4	-0.2	0.3	0.6

(a) Original Dense matrix

		0.2	-0.6		0.4		0.6
0.4		0.4			-0.4		0.5
0.7		-0.3		0.5		0.5	
	0.6	-0.5		-0.4			0.6

(d) Bank-balanced sparse matrix by local pruning inside each 1x4 bank

任务一：确定Compression的Search Space

1. Bank的大小
2. 每行的剪枝率

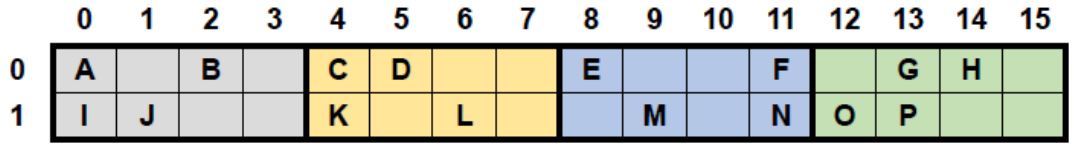
（如何确定？）

		0.2	-0.6		0.4		0.6
0.4	-0.3	0.4		0.2	-0.4		0.5

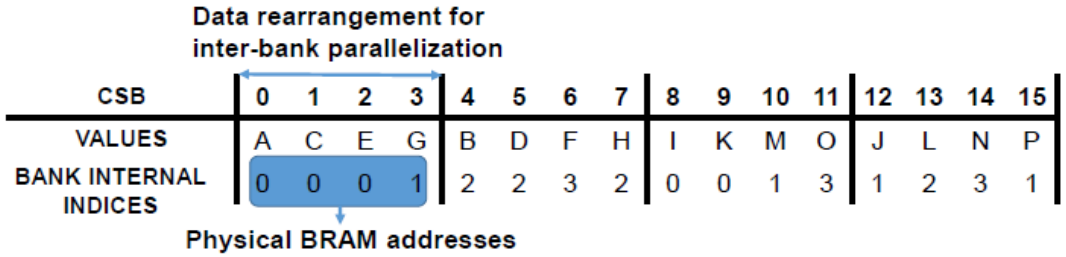
任务二：计算BBS’的存储方式带来的开销

压缩的稀疏矩阵存在BRAM中可以进行并发访问，密集矩阵(input)存在DRAM上进行随机访问。

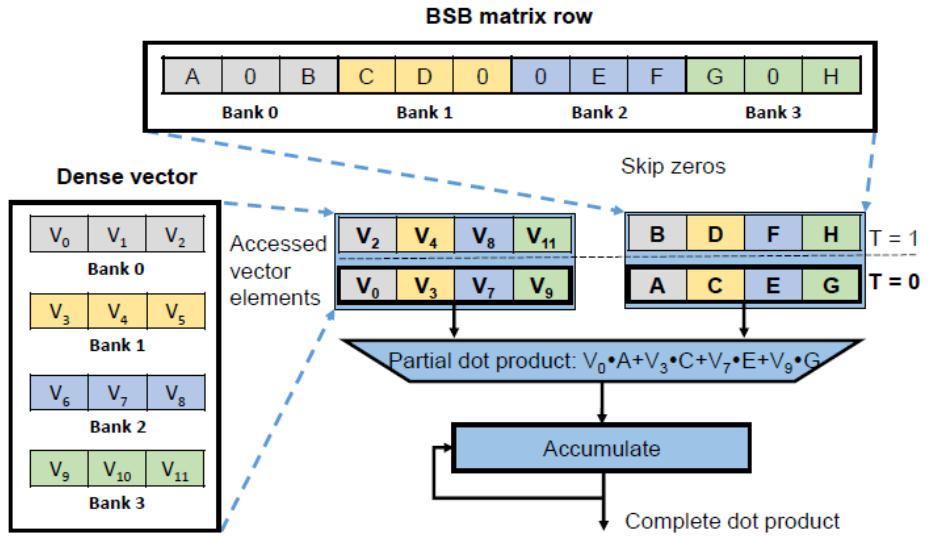
- 存储：
  1. 减少了被剪枝的数据量：$\sum_{i=0}^{R} C \cdot Rate_i$
  2. 但是增加了索引值的存储：$\sum_{i=0}^{R} C \cdot (1 - Rate_i)$
- 时延：
  1. 数据(ifm, pruning weight)在off-chip/on-chip memory的传输时延。
  2. 将要计算的数据Load到buffer中的时延
  3. 计算
  4. 将结果输出



(a) Original densely represented matrix



(c) CSB represented matrix



为什么CSB对硬件是友好的？

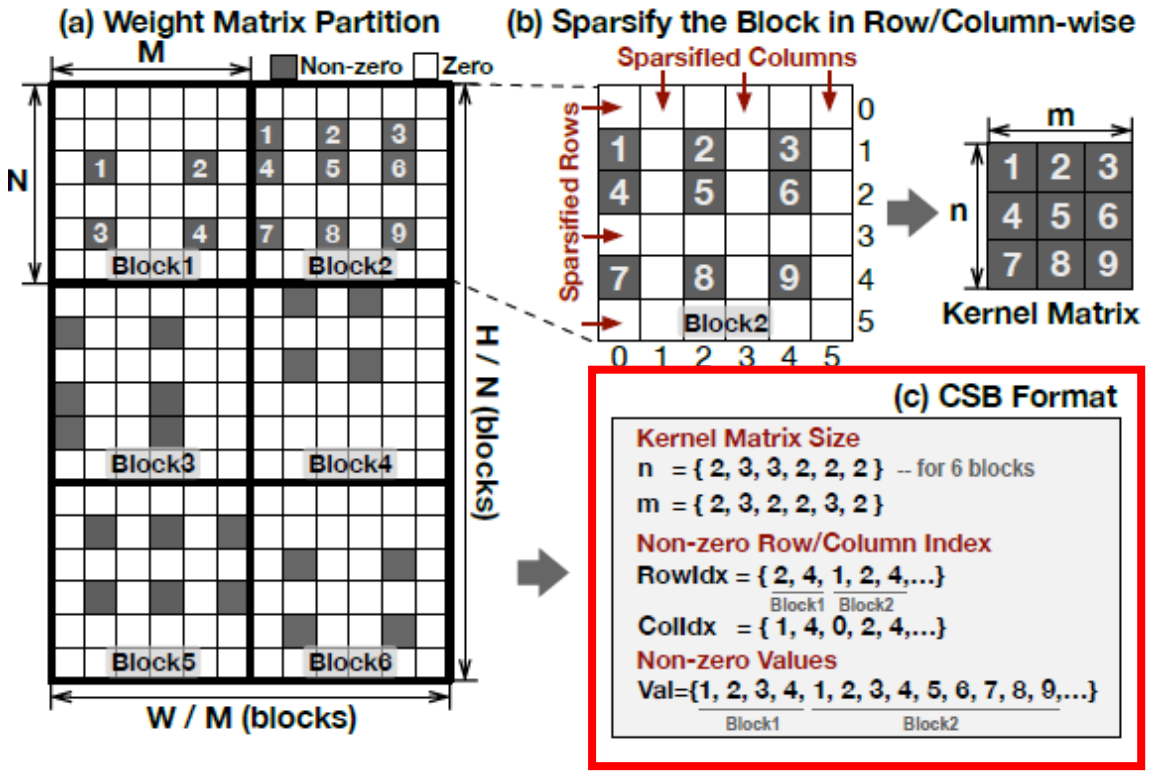
CSB Format:

n&m: 每个块内非稀疏行和列的个数；

RowIdx&ColIdx: 每个块非稀疏行和列的索引号

Val: 每个块内的非零权重的值

根据n、m、RowIdx、ColIdx就可以构建出一个密集矩阵，且因为计算过程中对这些稀疏块的访问是顺序的，因此省去了任意访问时对偏移量的计算。



下周计划

1. 本周未完成任务三，下周需要先建立transformer的performance model，然后建立稀疏之后的performance model
2. 如何剪枝？
3. … …