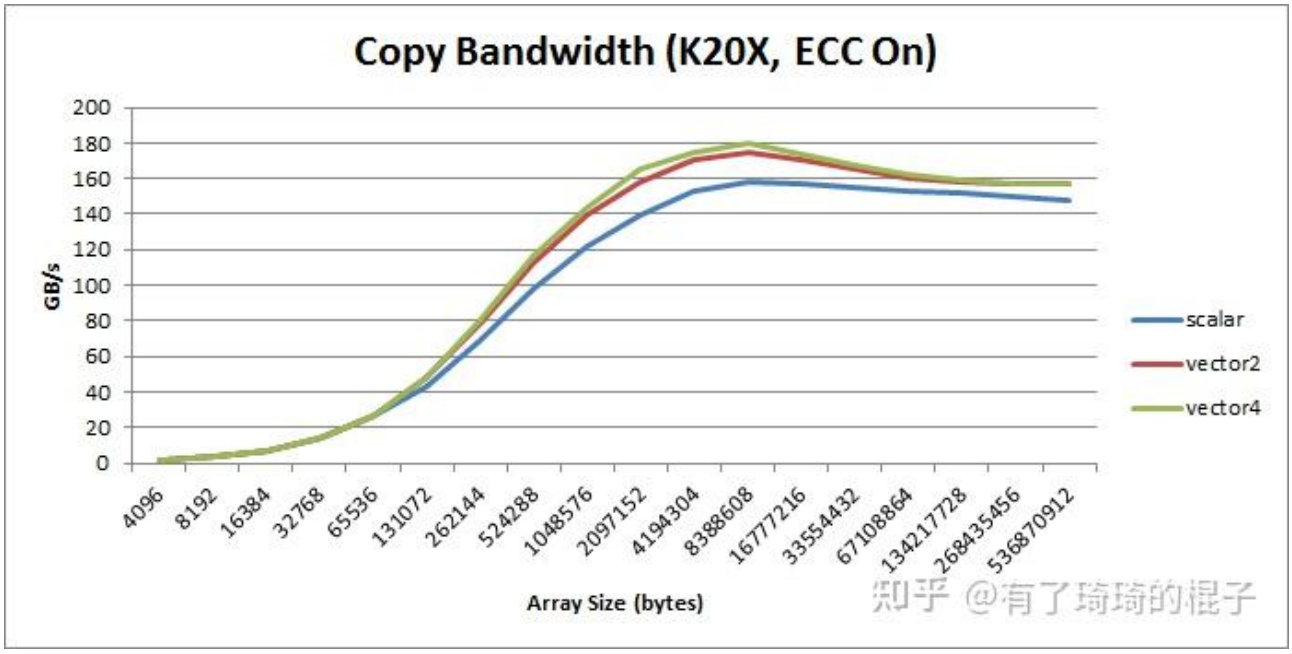
cuda编程中，转为float4是什么?

因为CUDA并没有向量计算指令，所以float4的使用主要是为了访存。为什么要转成float4访存更快，这个其实在nvidia的博客中有了详细的说明。参考https://developer.nvidia.com/blog/cuda-pro-tip-increase-performance-with-vectorized-memory-access/ 。

其核心就是**减少了一半的访存指令**，而为什么减少了一半的访存指令能够带来性能提升，这个逻辑是怎么来的。我理解主要有两点：

1. 对于访存单元而言，需要发射的访存指令更少了。假设用的是正常的float，那么读取4个float的话，需要发射4条LD.E指令。而使用float4的话，只需要发射一条LD.E.128指令。那直观上性能自然更优。但需要说明的是，在Nvidia的GPU中，由于SIMT架构是通过切换warp来掩盖访存的延时，所以并不代表着4个cycle发射4条指令就比1个cycle发射1条指令慢4倍，大部分的时间其实都是在等待访存单元把数据拿回来，而真正访存的时间，不管是去L1还是L2拿数，cache line都是128Byte。float和float4都是一样。对于一个warp而言，如果32个线程想要去拿128个数，不管float还是float4，都得变成4次对cache line的读取，当然，如果仅仅是拷贝，这个cache line还不一定命中。如果到了global mem中去读，从硬件的角度而言，访存端口也是一样，并不会因为float4就能够获得更多的端口读数。所以结论是，float4会更快，但是快得不多。



标量化与向量化数据拷贝性能

从实验里面也可以看出，开始段和结尾段，相差不多，相差最大的部分是中间段，float4有180GB/s，float160GB/s，但是为什么这个数量下的数据拷贝相差这么大，而别的数量下相差不多，这个现象如何解释，并不明确。

2. 对于指令cache而言，所需要的指令更少了，那么icache不命中的概率就会减少很多。我们假设一个场景，对于一段核心代码，volta架构有12KB的L0 指令cache，一条指令需要128bit，那么最多可以容纳768条SASS指令，对于sgemm中的核心循环，假设取12行12列，组成12x12=144条FFMA指令，循环展开6次，144x6=576条指令，一共有(12+12)*6 = 144个数需要load，如果用float则需要144条指令，那么计算和访存一共有720条指令，再加一些其他的指令，很容易导致指令cache放不下，性能有所损失。如果用float4的话，则需要144/4 = 36条指令，总共612条指令，指令cache肯定能放得下。

当然，转成float4也会产生一些负面的影响，首先是所采用的寄存器更多了，寄存器资源被占用多了之后，SM中能够并发的warp数量会有所减少。此外，如果本身程序的并行粒度就不太够，使用float4的话，所使用的block数量减少，warp数量减少，性能也会有一定的影响。所以如果是并行粒度本身不太够的情况下，还是需要谨慎地考虑是否采用float4这样的向量化数据。