

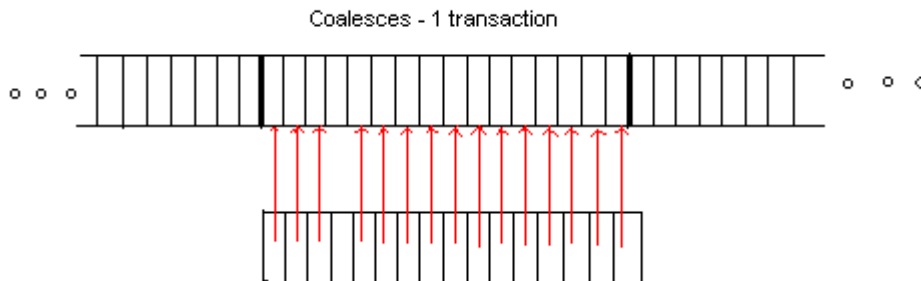
How to efficiently use global memory ?(Coalescing issue)

One of the important consideration in terms of global memory accesses is memory coalescing. Coalescing is a result of global memory access of 32, 64, or 128- bit words by half wrap of threads. And also, all 16 words must lie in the same segment of size equal to the memory transaction size.

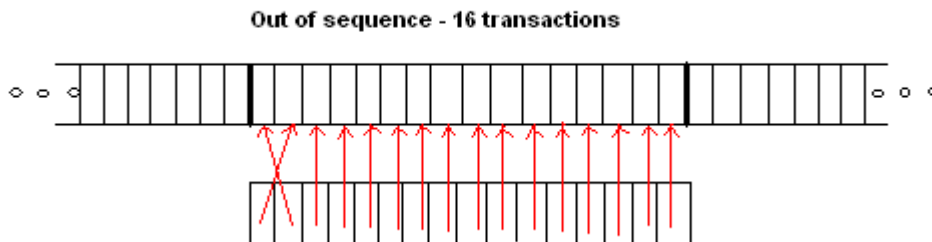
The way on which coalescing is done depends on compute capability. In the given pictures, taking example of float(32-bit)data example.

Coalescing for compute capability 1.0 and 1.1 :-

For compute capability 1.0 and 1.1, k-th thread must access k-th word in the segment(or k-th word in 2 contiguous 128B segments for 128-bit words) but not all threads need to participate.

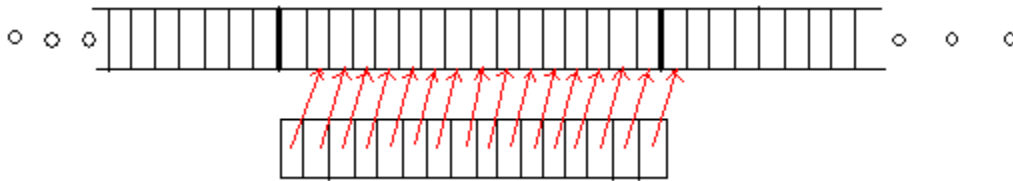


(1) In first picture, k-th thread accesses k-th word in one segment and 4th thread is not participating. Therefore it requires one 64B transaction i.e this global memory access is coalesced.



(2) In second picture, threads are access words of memory out of sequences, so, instead of one single 64B transaction here hardware splits data into 16 different transactions each for individual element. Since there is minimum transaction 32B so here we are effectively wasting large percentage of each memory transaction. Hence, in this case global memory access is not coalesced.

Misaligned - 16 transactions

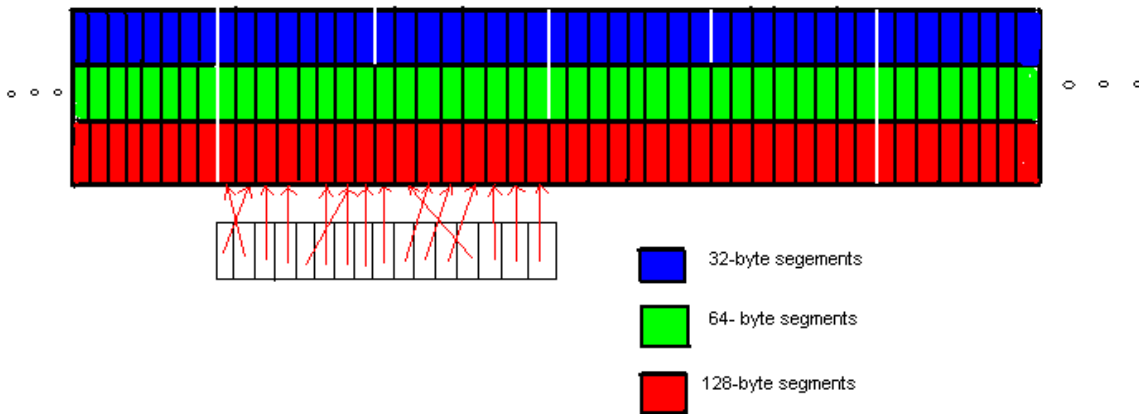


(3) In third picture, the case of misaligned memory access. Here continuous threads access continuous location of memory but the alignment is not correct.

Coalescing for compute capability 1.2 and higher :-

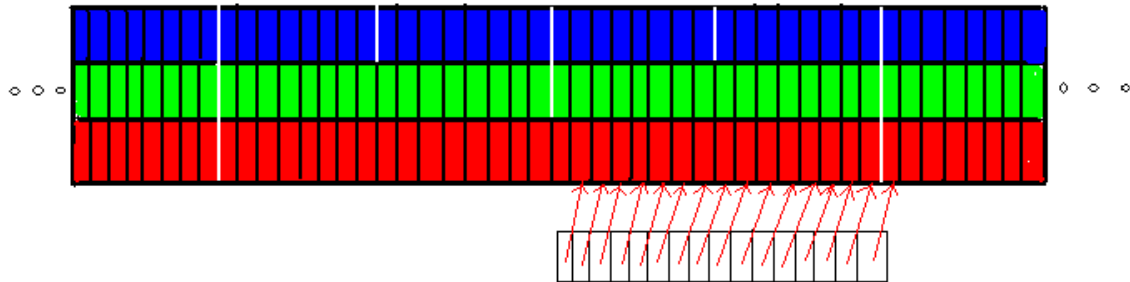
Device with compute capability 1.2 and higher issues transactions for segments of 32B, 64B, and 128B. So, smaller transaction should be avoided to utilize bandwidth.

1 transaction - 64B segment



(1) In fourth picture, all 16 threads of a half wrap access out of sequence of memory location in memory but all those locations lie in same 64B segment. So, memory system just perform one 64B transaction.

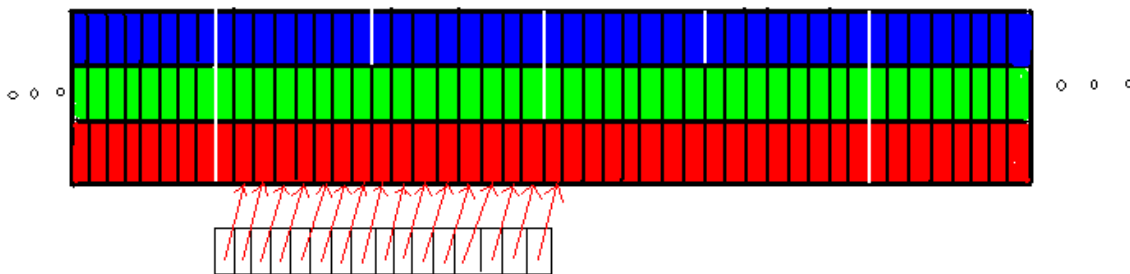
2 transactions - 64B and 32B segments



(2) In fifth picture, here is the case of misaligned memory access and continuous threads access continuous location of memory but crossing 128B segment boundary. Therefore here two 128B transaction required but in device reduces the transaction size, since in first 128B segment upper 64B is used so device reduces it to one 64B segment and similarly in second 128B segment device reduces it size to 64 B segment but further it reduces to 32B segment because upper 32B is not used.

Hence , Finally here two transactions are performed 64B and 32B.

1 transaction - 128B segment



(3) In sixth picture, all 16 continuous threads access continuous memory locations and also all memory locations lie inside same 128B segment. First 15 threads lies lower half of this segment and one thread lies upper half of this segment, so, this required one 128B transaction.
