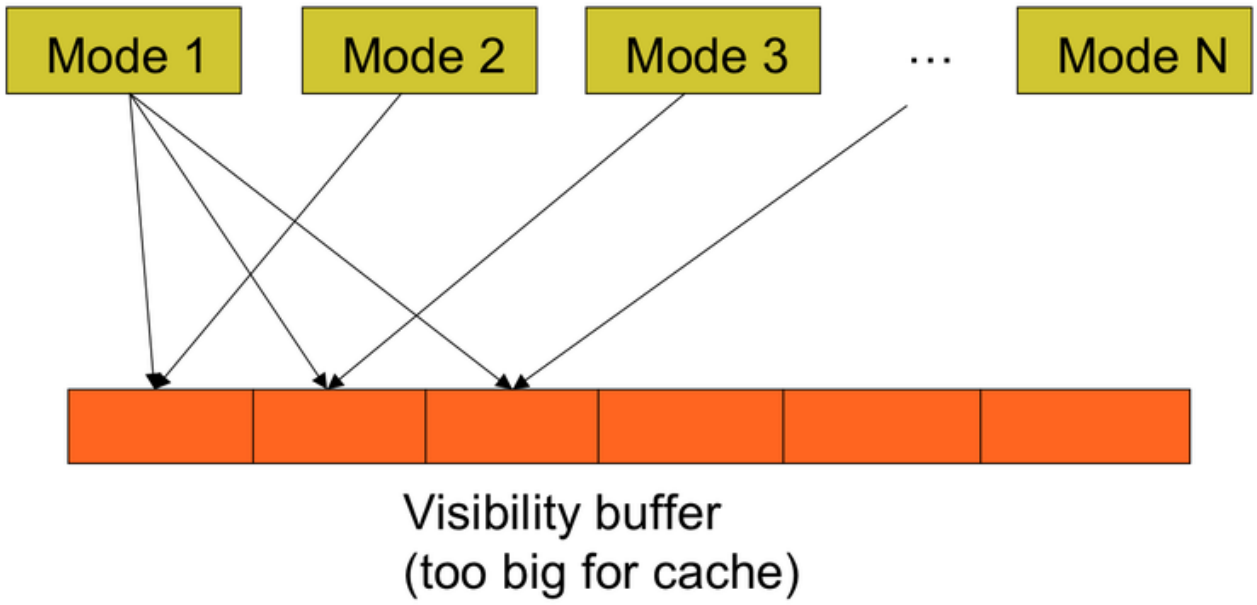
Baseline-based efficiency improvements in DiFX2.0

DiFX1.5 traversed the station spectra and cross-multipled to form visibilities in a rather naive way: all polarisation pairs of all frequencies of the first baseline were computed, then all pols of all freqs of the second baseline, and so on. This approach is illustrated in the diagram below:

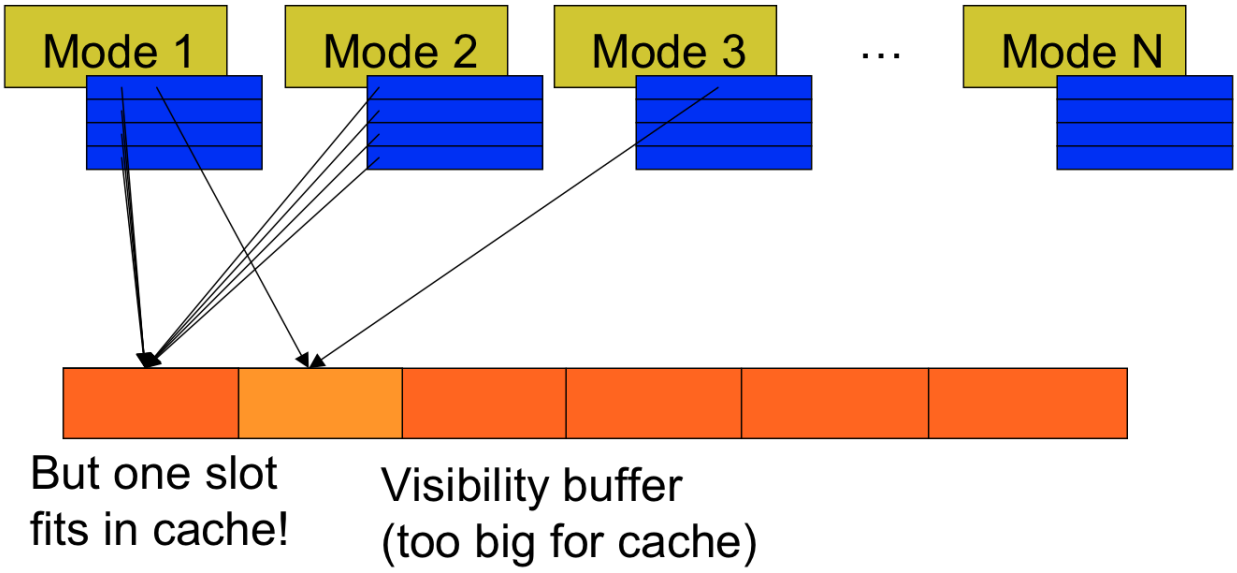


This is fine for small numbers of antennas/small numbers of channels, since the entire visibility result vector could sit in cache. However, going to larger numbers of antennas and especially larger number of channels can make the visibility vector too large to fit in cache. What happens then is that by the time the entire array has been traversed, the start of it has been overwritten in cache. The next time the correlator attempts to cross-multiply, the start of the results vector must be pulled back into cache. This in turn bumps out the middle of the result vector, and so on. Basically, nothing lives in cache for more than one cycle. This is bad news! Waiting on cache misses kills the correlator performance - DiFX1.5 sees factors of >5 slowdown as soon as you blow the cache limit (about 512 channels/band for 10 stations, 256 Mbps).

DiFX2.0 mitigates this problem by buffering multiple FFT results for each station before starting the cross multiplication. Additionally, the cross multiplication itself is divided into manageable chunks of channels, rather than trying to do too many at once (ensures the important things stay in L1 cache). Together, this ensures that:

- The intermediate vectors for fringe rotation, fractional sample correction etc for each Datastream stay in L2 cache for the duration of the buffering; and
- The result visibility vector stays in L1 cache for the duration of the cross-multiplication after the buffering.

The following diagram illustrates what I mean:



Say you buffer 10 FFTs at each station. That means you only get a cache miss for 1 of every 10 cycles, rather than every single cycle. This is little enough to make the cache missing a relatively small addition to the total processing increase. (There are unavoidable non-linear increases in processing cost going to either more stations or more channels. With more stations, you have more cross-multiplications to do, since there are more baselines. This starts to get significant at around 10-15 stations. With more channels, the cost of the FFT is going up with log N. Since the FFT is now dominating in DiFX2.0, that means going from 128 channels to 2048 [x16, = x4 cost to FFT] is likely to hit you pretty substantially - a factor of 2-3 at least in station-based cost.)