This paper discuss about the technique to decrease miss penalties in shared-memory multiprocessors by using hardware regular stride techniques, that is, sequential prefetching to prefetch several consecutive blocks following the block that misses in the cache.

**Strengths:**
1. A useful technique to exploit spatial locality. It can efficiently reduce the number of read misses, read penalty, and the execution time when applications exhibit good spatial locality.
2. It provides relative simple and low implementation cost hardware to dynamically detect access strides.
3. The adaptive sequential prefetching is a good scheme to reduce the cold miss and true sharing miss without increasing the false sharing miss component.
4. It can sometimes reduce the false sharing misses because a block that has been invalidated because of false sharing can be subsequently prefetched.

**Weaknesses:**
1. The blocks prefetching is done one by one. And every time before a block to be prefetched, a lookup that to see if the block is already in the cache should be done first. So, when the degree $K$ is large, the lookups are time consuming, power consuming, and cause memory traffic. Also, if a processor requests another block that not resides in cache during prefetching, it should wait until the prefetching finish and thus increase more traffic.
2. In the first paragraph of Page 737, the way to calculate prefetched blocks using only one PrefetchBit may not enough. Since a block may be referenced more than once due to temporal locality, the UsefulCounter indicate the number of blocks referenced, but not the number of **different blocks**. So UsefulCounter may not indicate the real numbers and affect the calculation of $K$. I think maybe the PrefetchBit should not be cleared.
3. There are some programs that have irregular data access and exhibit poor spatial locality, such as graph problem-solving, and event-driven simulation programs. So the techniques in this paper may not useful when those programs are running.
4. Since cache size is limited, prefetching may displace useful cache blocks from the cache and cause new cache misses. And it is most hazardous to performance when cache size is small and block size is large.