“Sequential Hardware Prefetching in Shared-Memory Multiprocessors”
Summary of Srivatsan Chellappa’s paper presentation in class and the critiques.

This paper discusses in detail how sequential prefetching technique (hardware controlled) can be adopted to reduce read miss penalties on processor utilization in shared-memory multiprocessors. The paper gives a detailed performance analysis of a class of hardware-based prefetching schemes for shared-memory multiprocessors and a simple adaptive version of this scheme.

The paper proposes 2 different hardware sequential prefetching techniques: **Fixed Sequential Prefetching** and **Adaptive Sequential Prefetching**. Experiments were conducted using test benches to analyze the effects on misses and traffic of fixed sequential prefetching, adaptive prefetch vs. fixed prefetch and no prefetch, dynamic behavior of the adaptive prefetching scheme, effects on total execution time, effects of finite size secondary level caches, processor speed and network bandwidth variations.

**Strengths:**

1. The author presents good reasons for the need of sequential prefetching, providing adequate insight into necessary background topics such as impact of block size on access penalties and traffic.
2. Sequential prefetching takes advantage of spatial locality to reduce the read stall time
3. Schemes are non-binding, so, they are applicable to standard processors with blocking loads.
4. Unlike software-controlled prefetching, the processors do not need to support specific prefetch instructions.
5. Number of cache lines prefetched can be varied dynamically.

**Weaknesses:**

1. In the paper, only K=1 (i.e. one-block look-ahead) was analyzed. If K > 1 was analyzed, we could get a better picture of the performance and also how much extra memory traffic is generated. It may lead to interference with other memory operations of the CPU and/or cause contentions and races during memory access.
2. The adaptive prefetching method will generate increased bus traffic in certain cases, again presenting the same problem as mentioned above.
3. Since the number of prefetches is dependent on the prefetching efficiency, and affects network traffic, applications with a high spatial locality will tend to suffer more from increased contention with the adaptive prefetching.
4. Since the local bus is clocked 3 times faster than the memories and since the adaptive scheme might send "bursts" of prefetch requests to the same memory module (because of the page-level interleaving) on read misses and so the input buffers of the memory module might have to be large.
5. The paper mentions that we increment a “use counter” for prefetched blocks to indicate usefulness of the prefetched blocks, and thus to adjust value of K. But it can also generate false sense of usage in certain cases. Say for example if 4 blocks are prefetched, and we keep hitting one of the prefetched blocks, the usage counter will increase indicating that prefetching 4 blocks was useful, but in reality only one block was used.
6. 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.