Proposed a new replacement and partitioning algorithms with a better balance between reuse and pollution

- Introduced a new concept, Protecting Distance (PD), which is shown to achieve such balance
- Developed single- and multi-core hit rate models as a function of PD, cache configuration and program behavior
- Models are used to dynamically compute the best PD
- Showed that PD-based cache management policies (PDP) improve performance for both single- and multi-cores

**Definitions**
- **Reuse distance**: The number of accesses to a cache set between two accesses to the same line
- **Reuse distance distribution (RDD)**: A distribution of observed reuse distances
  - A program signature for a given cache configuration
  - Is used to compute the best PD

**The Reuse-Pollution Balance**
- Predicting future access behavior is key to cache management
- Accuracy here is critical
- Consider 436.cactusADM from SPEC CPU 2006
  - Most lines are reused at 64 or fewer accesses
  - Lines should be kept in the cache for 64 accesses
  - Lines are not reused if evicted before that
  - Lines kept beyond that are likely to pollute cache
- The best RD thus balances reuse and pollution

**The Single-core PD P**
- Each cache line is assigned a value representing its remaining PD (RPD)
  - The RPD of a promoted line is set to max PD
  - The RPDs of lines in a set are decremented on access
- On a hit: The hit line is promoted
  - Selecting a victim on a miss
    - Line with an RPD > 0 is unprotected and can be replaced
    - Two cases when all RPDs > 0 (unprotected lines)
      - Caches without bypass (inclusive)
        - Replace unused line with highest RPD first
      - No unused line: Replace line with highest RPD
      - Caches with bypass (non-inclusive)
        - Bypass the new line

**Using a Static PD**
- SPDP-NB, SPDP-B: Static PD without and with bypass

**The Protecting Distance**
- The Protecting Distance (PD): the reuse distance where a majority of lines are reused
  - Lines are ‘protected’ until PD accesses to a set
  - Cannot be evicted during this time
  - Can be evicted after that if not reuse
  - Protection can be accomplished in a number of ways

**The Single-core Hit Rate Model**
\[
E(d_{d}) = \frac{\text{New Accesses}}{\sum_{i=1}^{n} (d_{i})} \times \sum_{i=1}^{n} \left(1 - \frac{d_{i}}{T} \right)
\]
- \(N_{e}(N_{i})\): The RDD
- \(d_{i}\): The protecting distance
- \(E()\): A function of RDD and PD
- The model is used to find the PD maximizing the hit rate

**PDP Cache Organization**
- RD Sampler tracks access to several cache sets
  - In L2 miss/WB stream, can reduce sampling rate
  - Measures reuse distances of a new access
- RD Counter Array collects \(N_{i}\) of accesses at RD\(_{i}\)
  - To reduce overhead, each counter covers a range
- PD Compute Logic finds PD that maximizes E
  - Computed PD used in next interval (5M L3 accesses)
  - Max PD = 256 in this work
  - Showed ways to use only 2 or 3 bits for RPD, not 8!

**Shared-cache Management**
- Thread-aware, each thread has its own PD
  - A thread’s PD balances its reuse and pollution
  - Replacement and bypass
  - A thread’s PD determines its cache partition
  - Its lines occupy cache longer if its PD is large
  - Using PDs, cache is implicitly partitioned per needs of each thread
  - Sampler and counter array replicated per thread
  - Compute a vector PD (Tp: number of threads)
    - \(E(\text{PDP}) = \sum_{i=1}^{n} \frac{\text{Miss}(F)}{\text{Access}(F) + T_{p}}\)
    - Proposed a heuristic to reduce complexity
      - A search algorithm is used to find a combination of threads’ RDD peaks that maximizes the hit rate
    - The single-core model generates top 3 peaks per thread
    - The complexity is \(O(T_{p})\)

**Evaluation**
- SPEC CPU 2006 Benchmarks, 1B instructions
  - Single-core: Compared to EELRU, DIP, SDP, RRIP
  - Multi-core: Compared to UCP, PIPP, TA-DDRIP
  - PDP-\(x\): PDP with \(x\) bits per line

**Table**
- Configuration Parameters
  - DL1 Cache 32KB, 8-way, 64B, 2 cycles
  - LL1 Cache 32KB, 4-way, 64B, 2 cycles
  - L2 Cache 256KB, 8-way, 64B, 10 cycles
  - L3 Cache (LLC) 2MB, 16-way, 64B, 30 cycles
  - Memory 200 cycles

**Graphs**
- 436.cactusADM: Breakdown of accesses and occupancy
  - Miss rate improvement for 436.cactusADM
    - Miss rate as a function of static PD
    - Peaks at 72, is very sensitive to how close it is to 64
    - Significantly better than existing policies
  - 436.xalancbmk: Comparing 3 execution windows
    - We target non-inclusive cache for the rest of the paper