

An Accurate and Detailed Prefetching

Simulation Framework for gem5

Martí Torrents, Raúl Martínez, and Carlos Molina

`martit@ac.upc.edu`

Computer Architecture Department

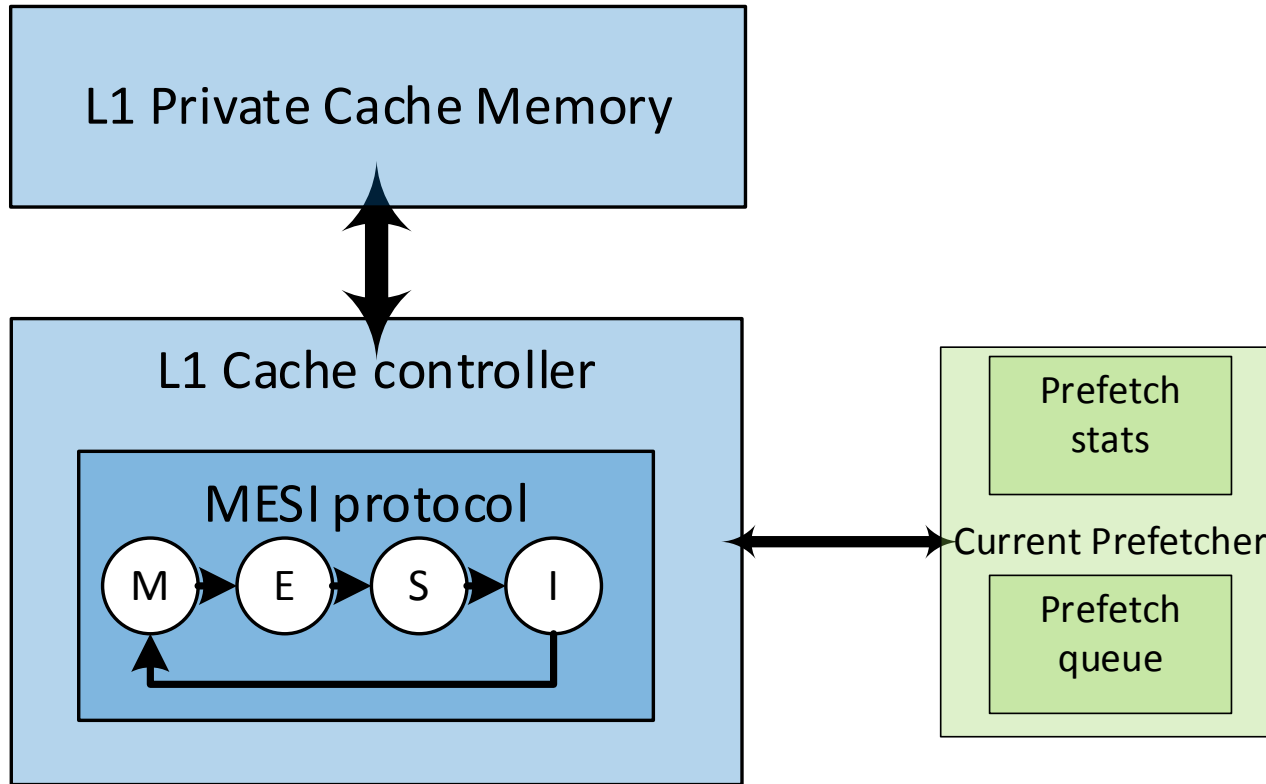
UPC – BarcelonaTech



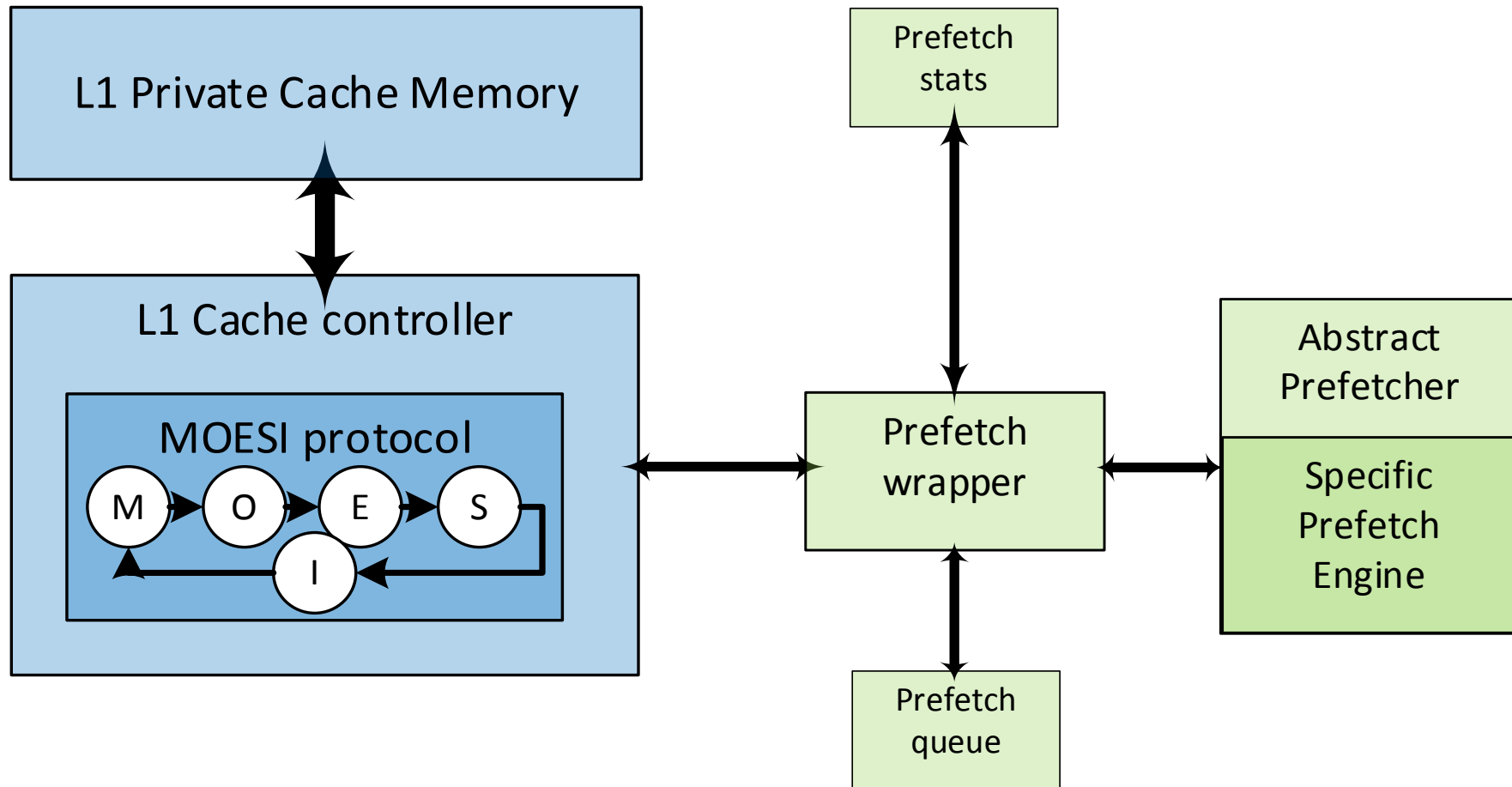
Prefetching

- Reduce memory latency
- Bring to a nearest cache data required by CPU
- Increase the hit ratio
- Implemented in many commercial processors
- Erroneous prefetching may produce slowdown
- Simulation tools should include this capability

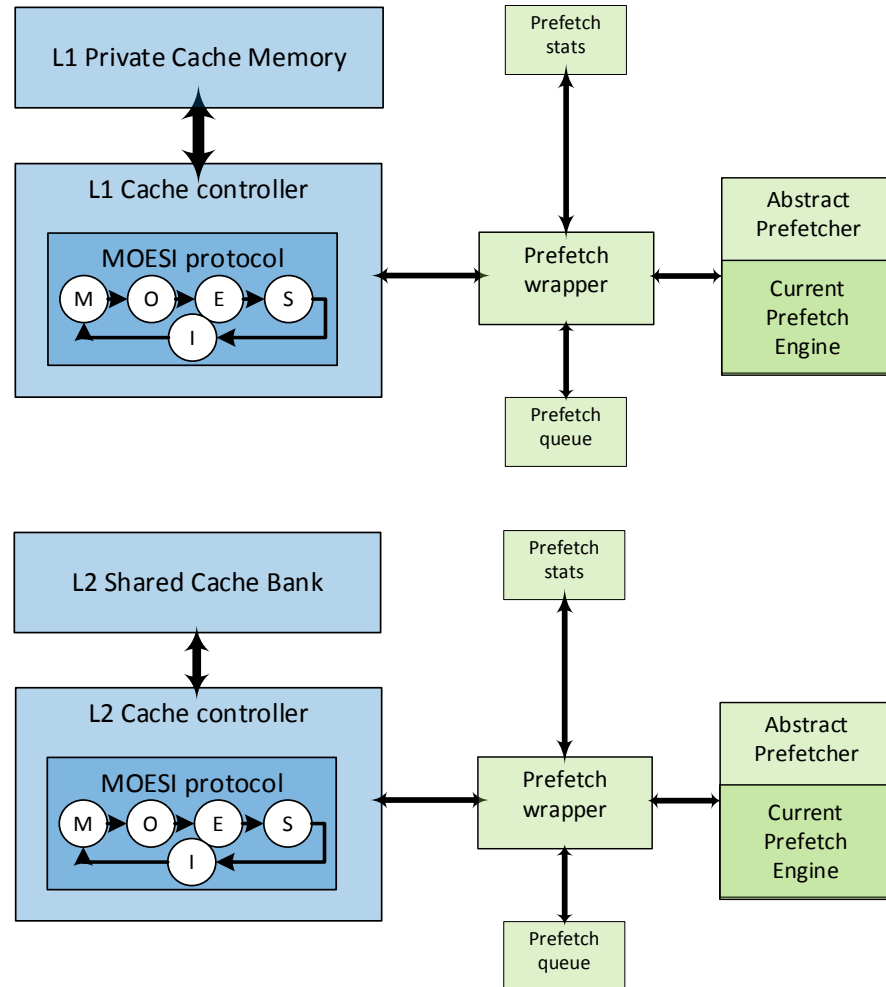
Current solution



Our solution

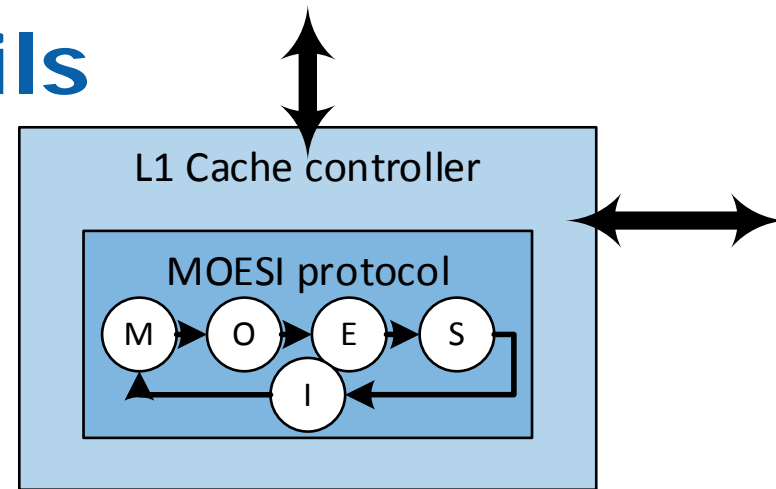


Our solution



Implementation details

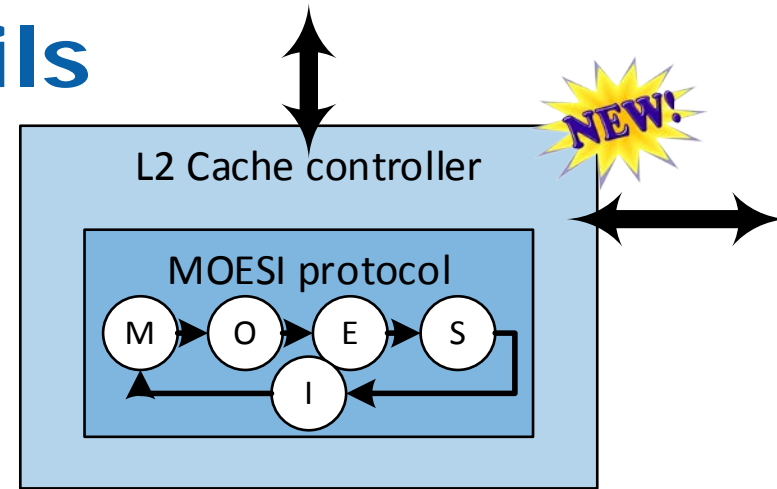
- Notifies the wrapper:
 - Cache accesses
 - Cache allocation
 - Cache evictions



- Reads from Prefetch Queue
- Prefetch issued in cycles without Loads/Stores
- Protocol modification similar to a Load operation
- Very similar to the current solution

Implementation details

- Same as in the L1 but...



- L2 local hits
 - Some data that is invalid in L2 but locally allocated
- L1_GETS does not store in L2
 - Protocol modified to store pref requests in L2
- Pref queue generates a request for another tile
 - Request is forwarded to the corresponding tile

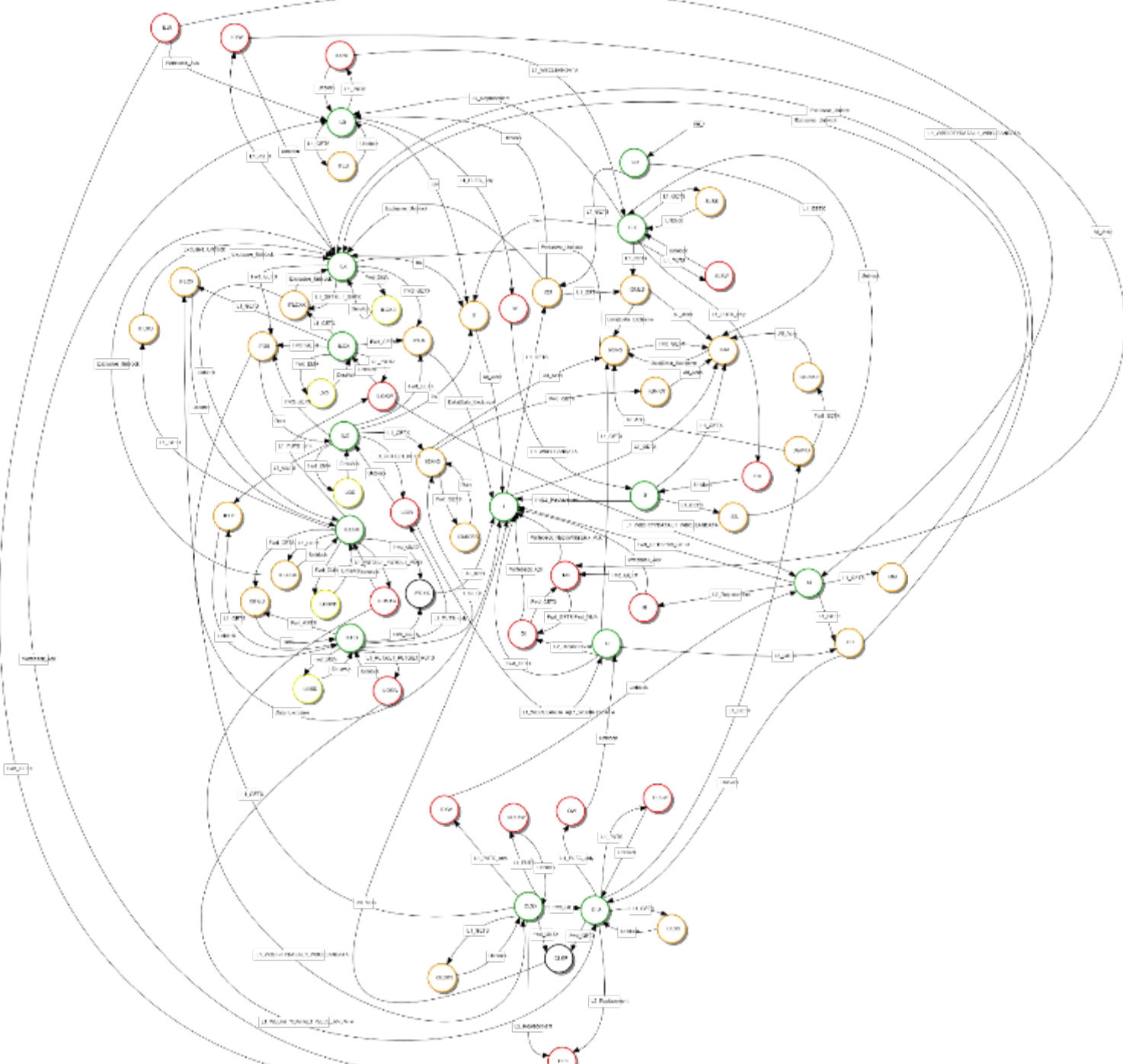
Im

• S

• L

• L

• P

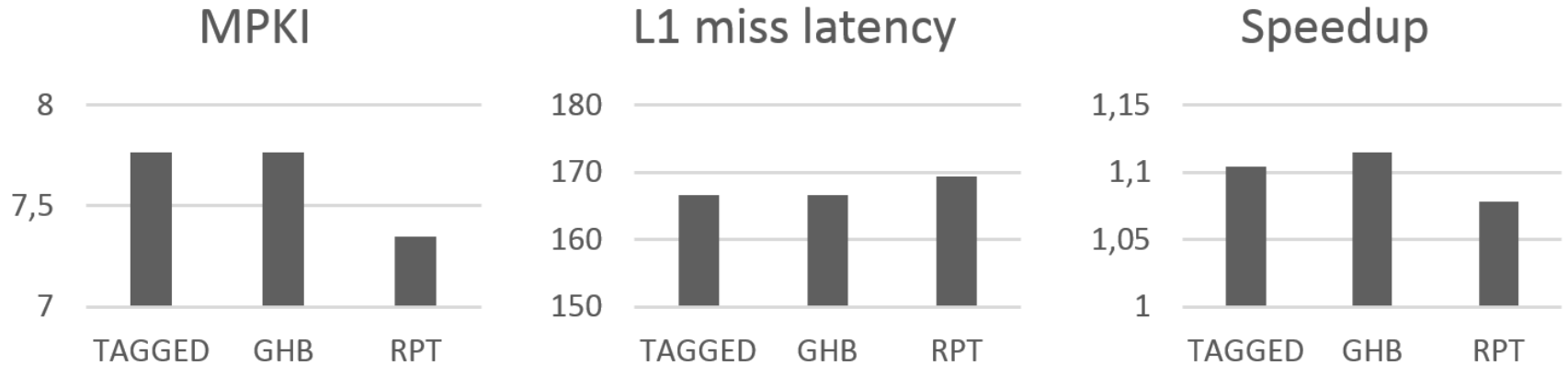


Case of Study: NoC aware prefetch performance evaluation

- We tested 3 classical prefetch engines:
 - Tagged Prefetcher
 - Reference Prediction Table (RPT)
 - Global History Buffer (GHB)
- With the gem5 Simulator using
 - 16 tiled x86 CPUs
 - L1 prefetchers
 - Ruby memory system
 - MOESI coherency protocol
 - Garnet network simulator
- Parsec 2.1



Case of Study: Results



M. Torrents, R. Martínez, C. Molina. "Network Aware Performance Evaluation of Prefetching Techniques in CMPs". Simulation Modeling Practice and Theory (SIMPAT), 2014.

Conclusions

- Prefetcher is important and it must be simulated
- Current solution is ok
- Our solution goes one step farther
 - Easy to change/add new prefetch engines
 - Detailed statistics about prefetching
- Current solution is ok for non prefetch researchers
- Current tool can be easily included in new solution
- Our tool is better for research related with prefetch

An Accurate and Detailed Prefetching

Simulation Framework for gem5

Martí Torrents, Raúl Martínez, and Carlos Molina

`martit@ac.upc.edu`

Computer Architecture Department

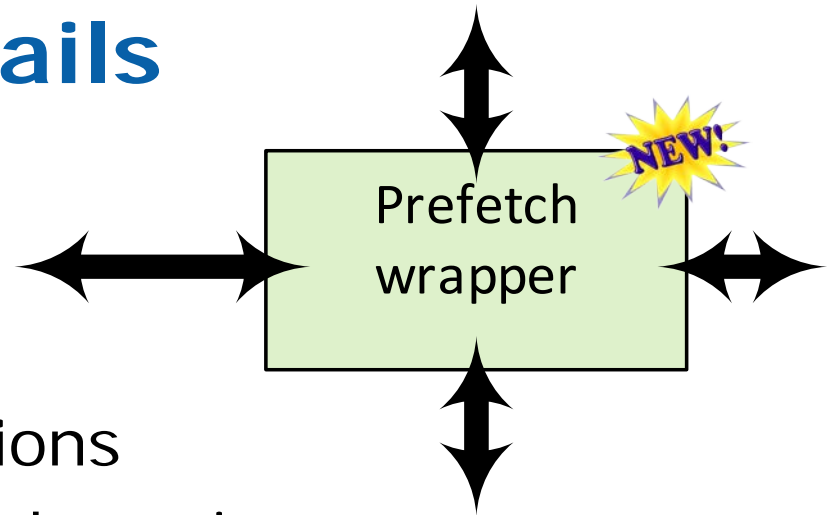
UPC – BarcelonaTech



BackUp Slides

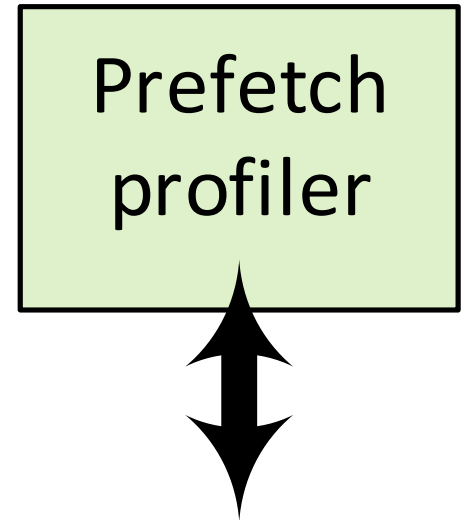


Implementation details










- Receives command line options
- Creates and inits the prefetch engine
- Manages communication Controller → Prefetcher
- And Prefetcher → Prefetch Queue → Controller
- Collects statistics

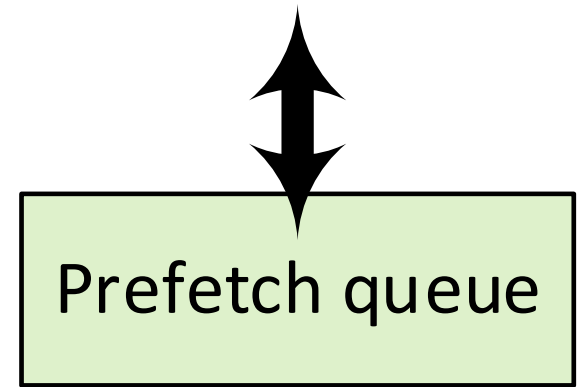
Implementation details




- Accumulates the statistics

- observed_misses - numMissesObserved
- Cancelled - numDroppedPrefetches
- completed - numPrefetchAccepted
- hit 
- in_cache 
- late - numPartialHits/numHits
- overflowed 
- page_faults - numPagesCrossed
- total - numPrefetchRequested
- unuseful 
- useful 
- queue_merged_requests 
- generated_prefetches_per_train - streams
- numMissedPrefetchBlocks 

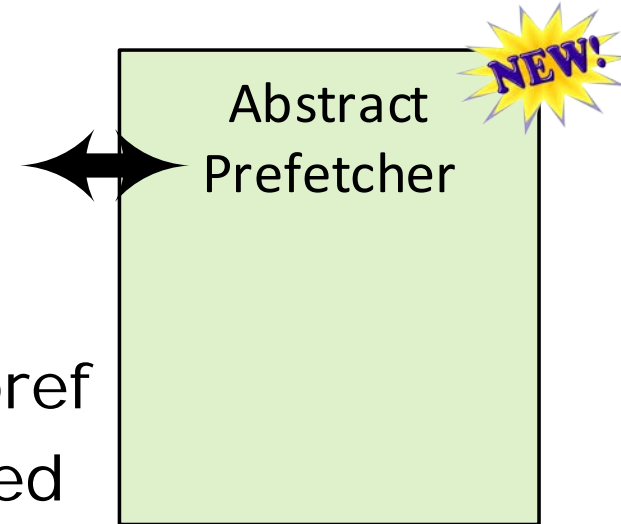
Implementation details



- Collects the requests generated by the engine
- Checks the page fault error or overflow
- Merges repeated requests 

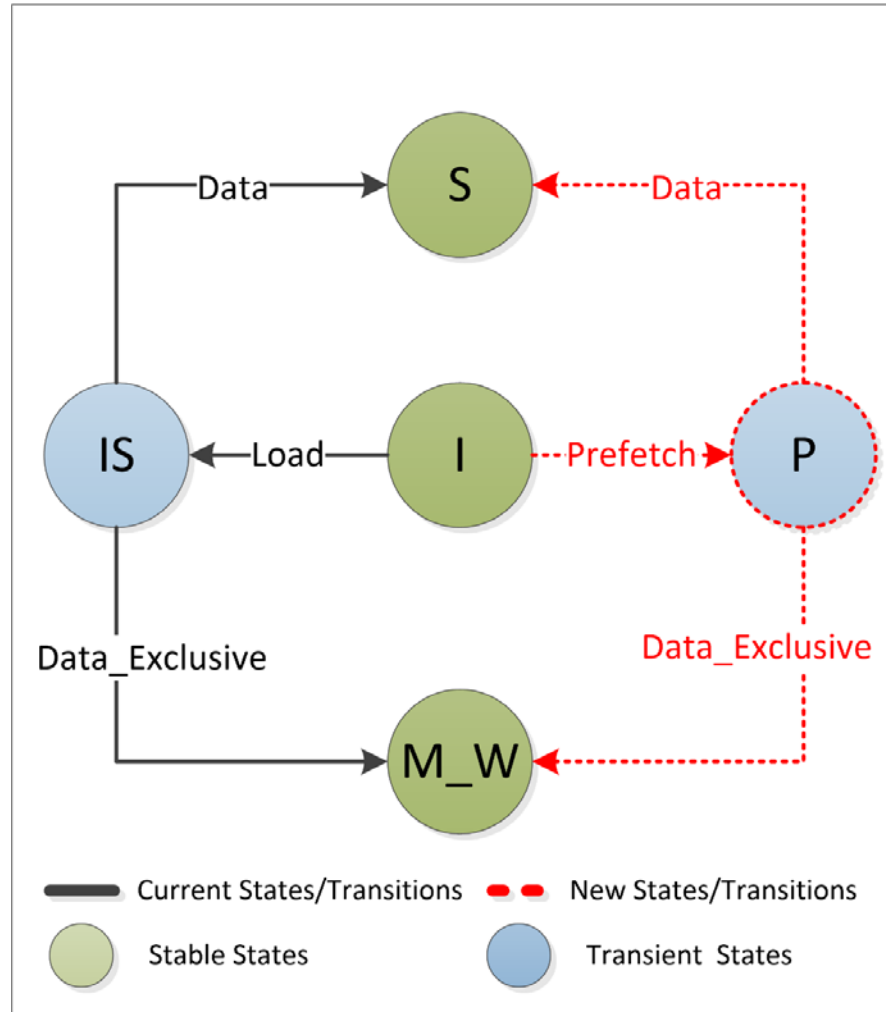
Implementation details

- Works as an abstract class
- Must be inherited by the specific pref
- Virtual functions must be redeclared



- Init: Initialization function
 - Prefetcher size, distance, aggressiveness, etc.
- Observe request: Called on each cache access
 - Hit/miss, prefetch/no_prefetch, accessed address, etc.
- Allocate: Called when data allocated in cache
 - Same as observe request
- Deallocate: Called when evicting from cache
 - Only evicted address

1st Level Cache Protocol Modifications



2nd Level Cache Protocol Modifications

