Simulating Snooping-Based Cache Coherence Protocols

Vishnu Razdan (vrazdan)
Zhaodong Zheng (zhaodonz)

The Project

- Easy-to-use tool for simulating caches
- Incorporate different cache protocols (MSI, MESI, MOESI)
- Usable on arbitrary executable files
- Accurately measure advantages/disadvantages of different code
- Atomic bus
- Snooping-based cache

Simulator Components

- **CacheController**[**Caches**]
- **Cache**[**CacheSets**]
- **CacheSet**[**CacheLines**]
- **AtomicBusManager** handles all **BusRequests**
- **Cache ->BusRequest**
- **Caches** snoop **BusRequests**

Inputs

- A binary executable
- A memory access trace file
- Cache configuration details

Outputs

- Cycles to complete
- Hits, misses, flushes, evicts
- Main memory requests, bus requests, cache-to-cache shares

Why?

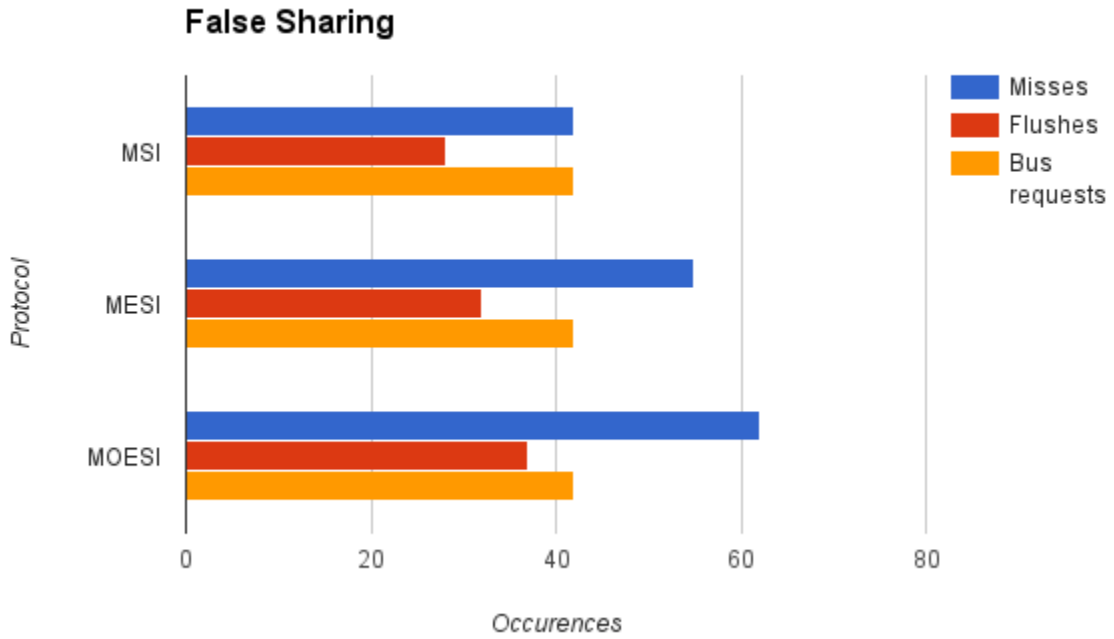
- Uncover problems
  - False sharing
  - Cache thrashing
- Compare different code implementations for performance
- Compare different coherent cache protocols

Why is this hard?

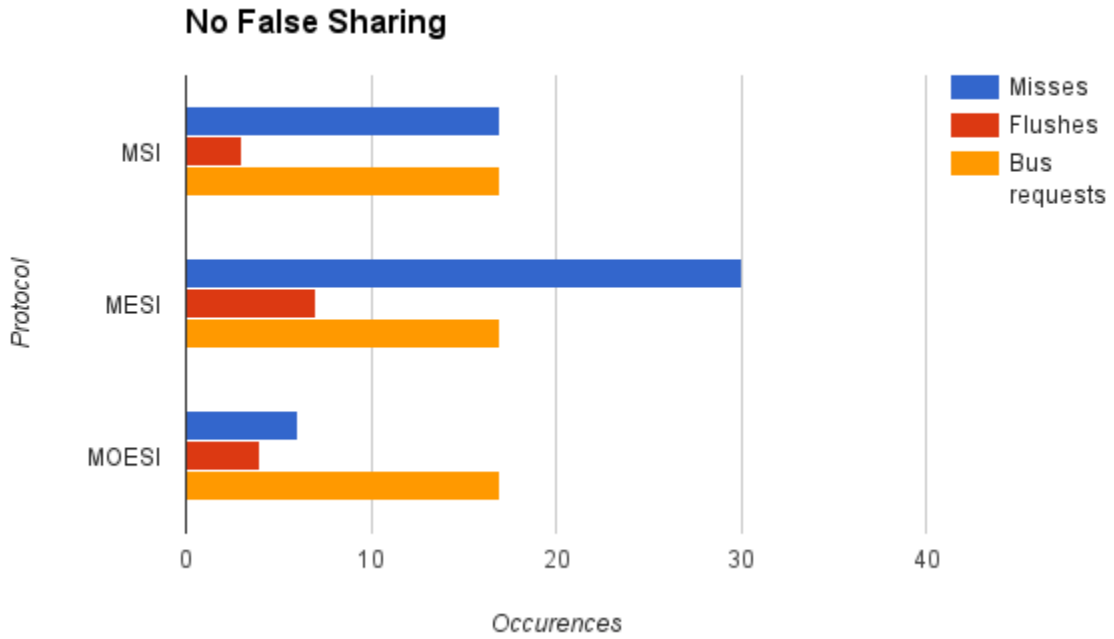
- Maintaining program memory operation ordering is difficult
- Correctly implementing the coherence protocols

Results

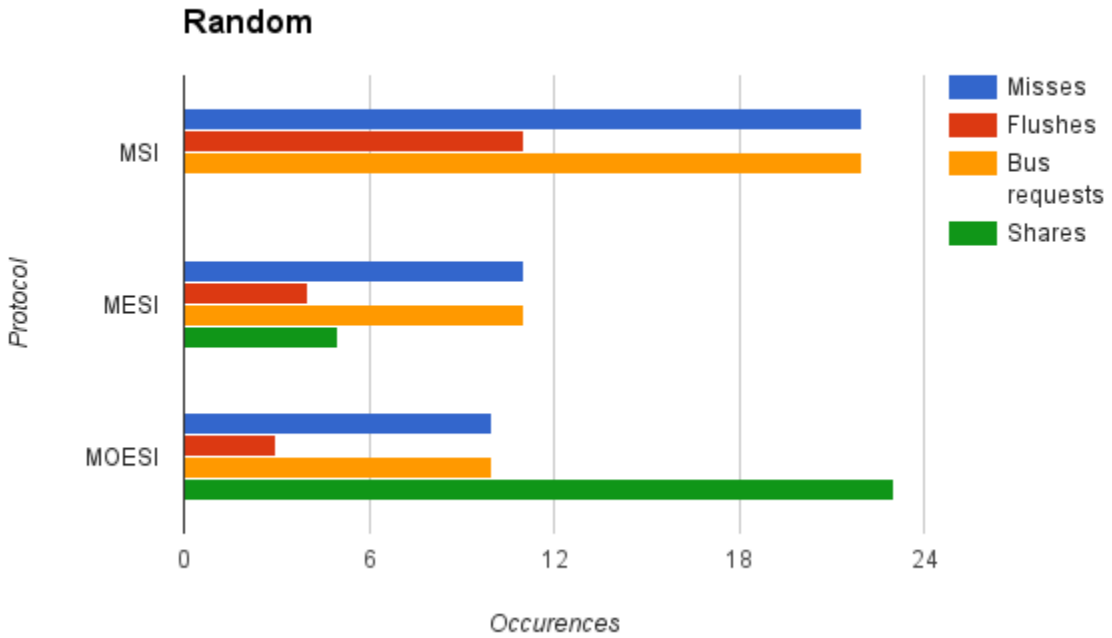
More complex protocols cause more overhead



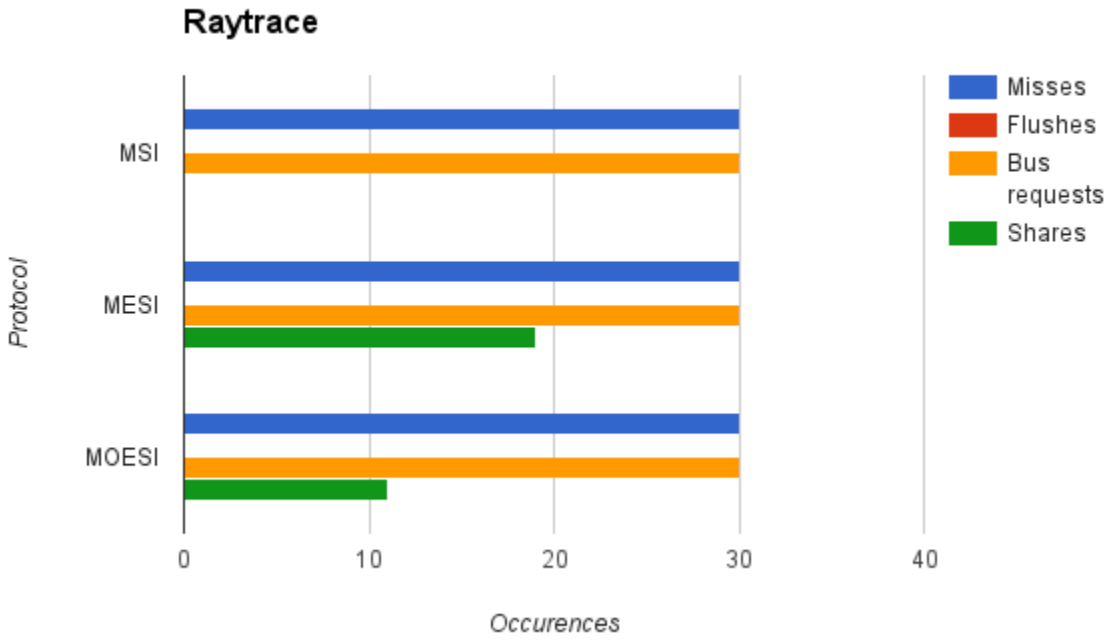
No false sharing allows MOESI to perform well



For random memory access, MESI and MOESI perform great

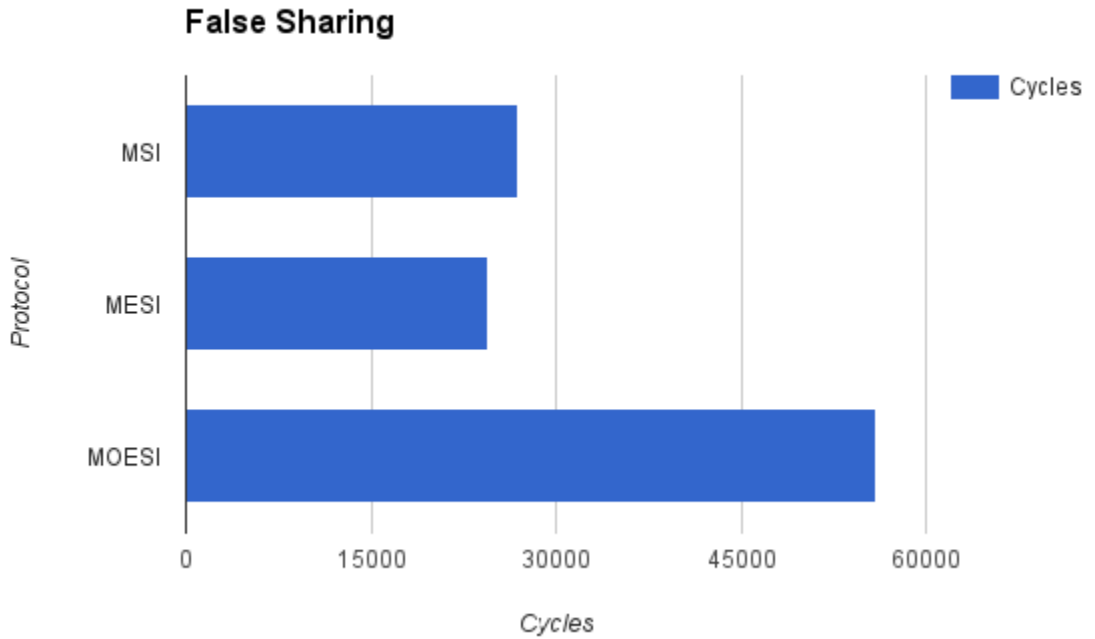


For certain real applications, there is no negligible performance gain

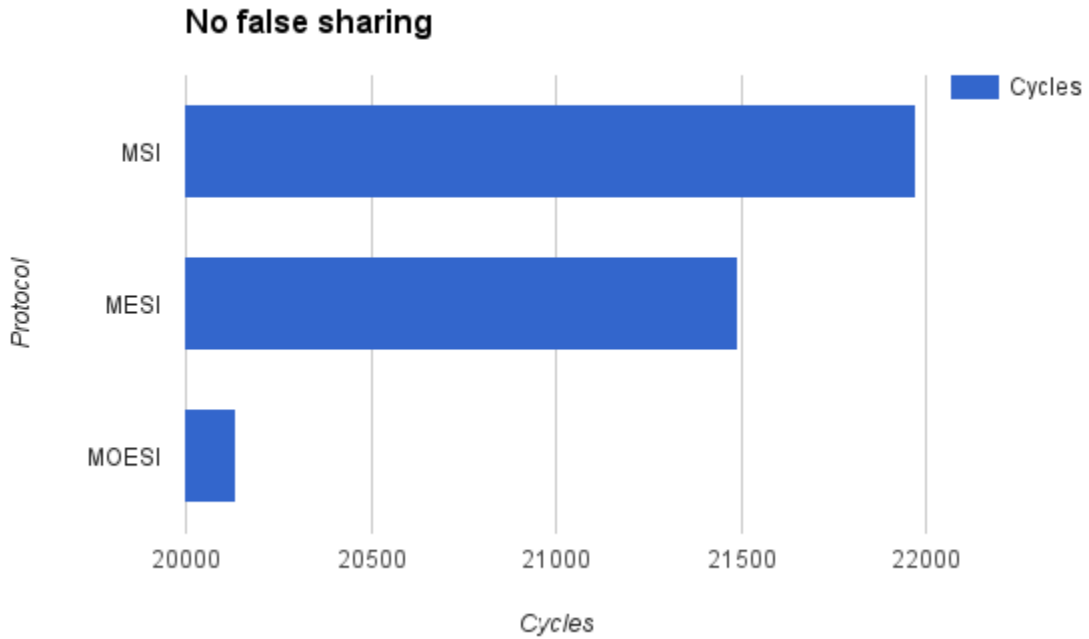


How the total execution time changes depending on what protocol used

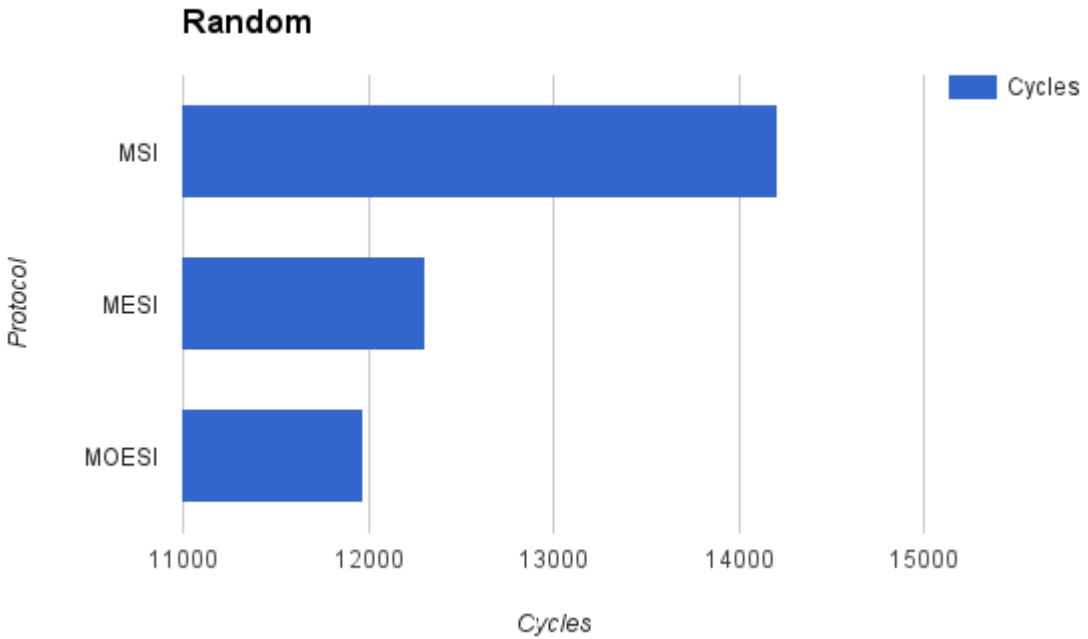
Cycle Cost per protocol when there is false sharing



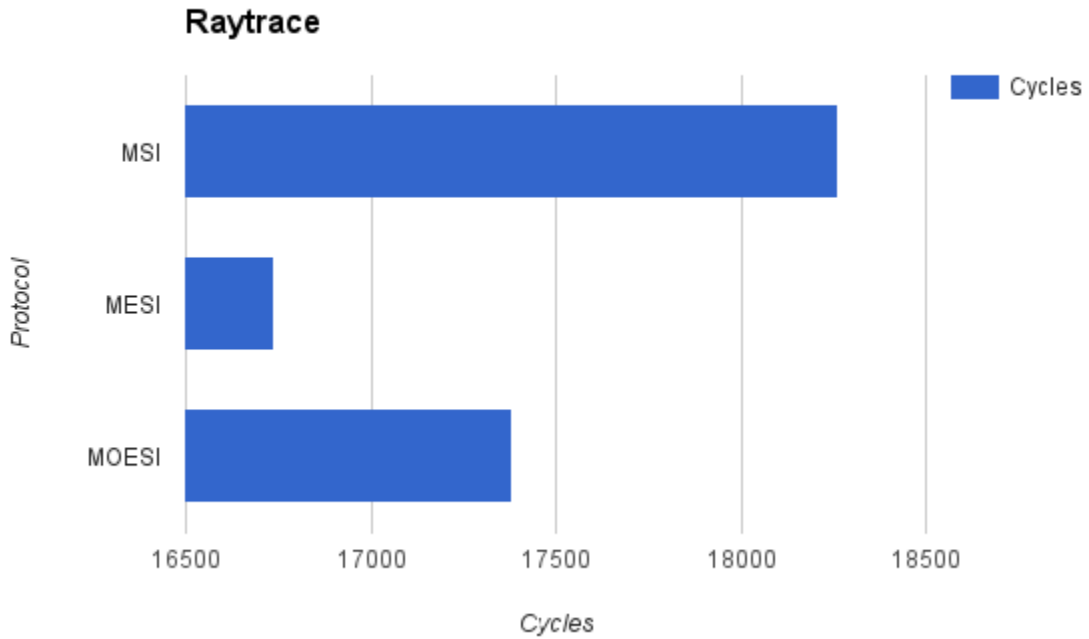
Number of cycles decrease with MESI and MOESI when there is no false sharing



Performance gained with MESI and MOESI protocols with random memory accesses



Performance gained with MESI for a raytracer



Thanks!