Memory Consistency Exceptions

18742-Computer Architecture and Systems

Nithin, Deepali Garg

Electrical and Computer Engineering
Carnegie Mellon University

February 18, 2020

Electrical & Computer ENGINEERING

Schedule

Electrical & Computer ENGINEERING

Overview

Overview

Similarities

1. Both the papers recognize that data races are hard to avoid, and racy programs are hard to debug
2. Both identify that region conflict detection is sufficient to identify data-races which can cause sequential consistency exceptions
3. Both relies on hardware dynamic detection of data races, in these synchronization-free regions

Differences

1. ***Conflict Exceptions***
   Doesn't care about the origin of data races, requires no additional compiler changes
   Stores meta-data per cache line
2. ***DRFx***
   Eliminates data-races due to compiler re-ordering
   Finite meta-data stored per region for conflict detection, in-turn constraints the hardware optimizations allowed per region

Electrical & Computer ENGINEERING

Conflict Exceptions

Motivation and Contribution

- Data races are everywhere.
- Methods to handle data races.
  - Define semantics to execute data race programs (causes unexpected behaviour.)
  - Make the programming languages such that data race is avoided while writing program itself.
  - Monitor the execution and handle the data races in run time.
- The existing happen-before race detection can detect data races but at a cost of time and space.
- Paper proposes a dynamic method to detect data races and raise exception,

Electrical & Computer ENGINEERING

Conflict Exception

- Divide the program into sync free regions.
- whenever there is a conflict in regions between two concurrently executing threads raise an exception.
- Advantages of Conflict Exception
  - Simpler Programming Language semantics.
  - Most debugging benefits of full data-race detection.
  - Recovery action support.

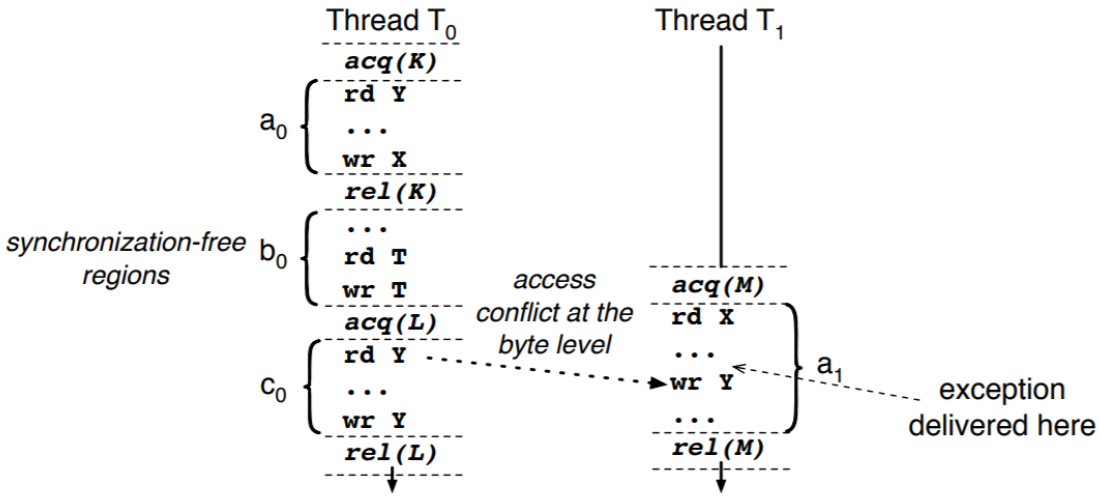


Figure: Example of Conflict Exception

Electrical & Computer ENGINEERING

Architectural Support for Conflict Exception

Hardware/Software Interface.

1. Regions are demarcated by special instructions: *beginR* and *endR*.
2. Instructions which are not in a region are singleton regions themselves.

Protocol State and Invariants

Name	What it records	Set on	Cleared on
Local read bits	Bytes read by the local thread during active region	Local read access (hit/miss)	End of region in local thread
Local write bits	Bytes written by the local thread during active region	Local write access (hit/miss)	
Remote read bits	Bytes read by other threads in their active regions	Local read or write miss	End of region in remote thread that originally set the bit
Remote write bits	Bytes written by other threads in their active regions	Local read or write miss	

(a) Access bit vectors associated with each cache line

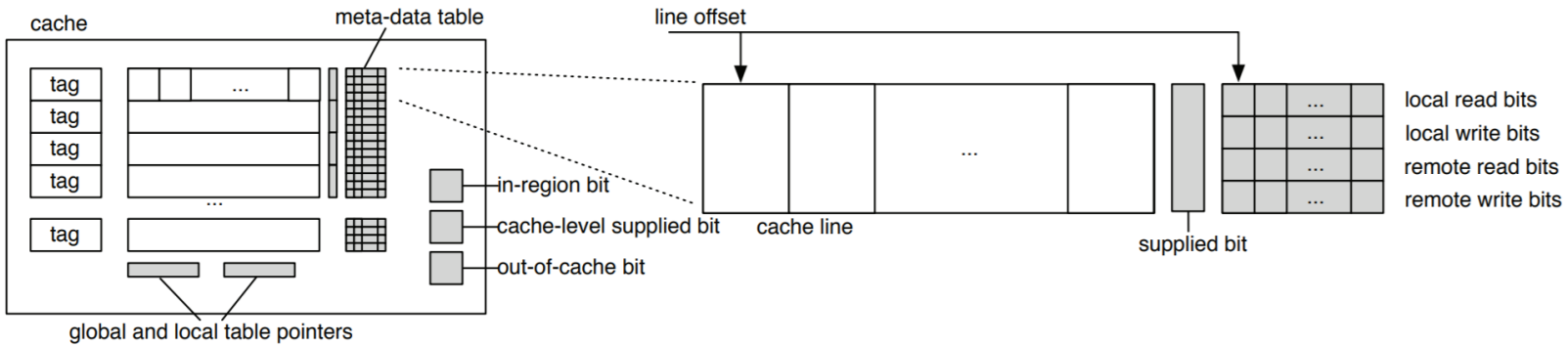
#	State	Type of bit set	Invariant
1	Any	Local read bits	A local read bit is set if and only if the local thread has read the corresponding byte within its active region.
2	Any	Local write bits	A local write bit is set if and only if the local thread has written the corresponding byte within its active region.
3	M or E	Remote read bits	A remote read bit is set in a cache if and only if at least one thread other than the local one has read the corresponding byte within its active region.
4	O or S	Remote read bits	A remote read bit is set in a cache *only if* at least one thread other than the local one has read the corresponding byte within its active region.
5	Any	Remote write bits	A remote write bit is set in a cache if and only if at least one thread other than the local one has written the corresponding byte within its active region.

(b) Invariants guaranteed by the protocol

Electrical & Computer ENGINEERING

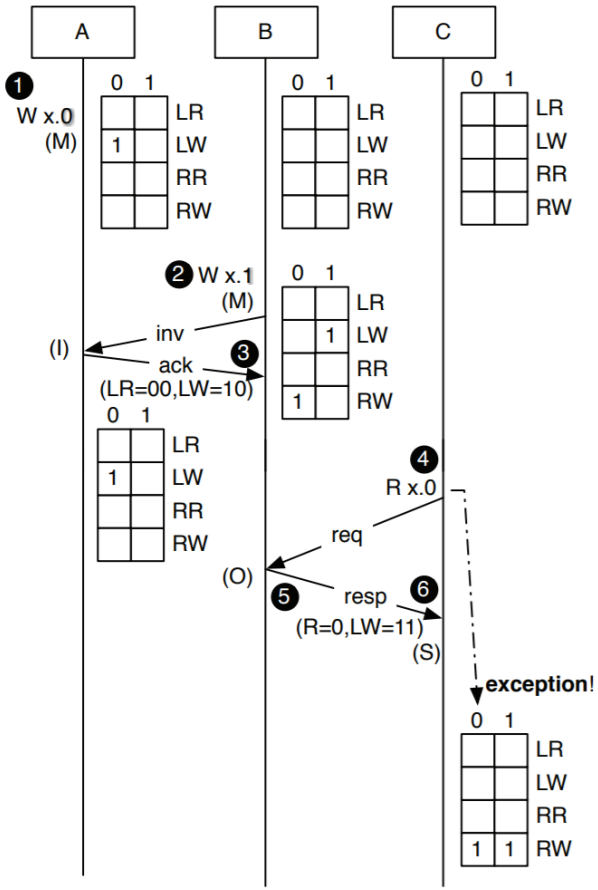
Architectural Support for Conflict Exception

Adding Support to the Coherence Protocol

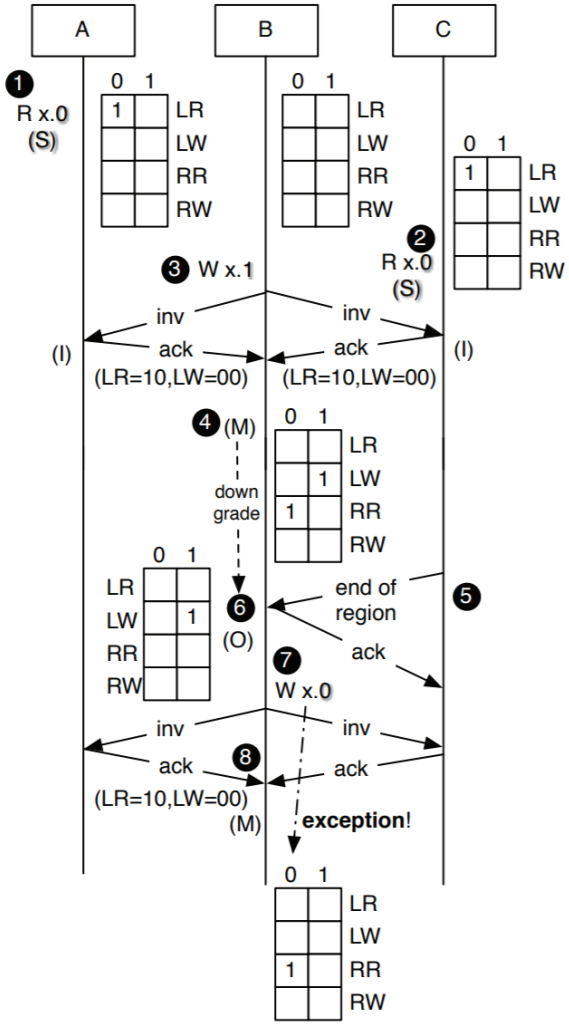


Architectural Support for Conflict Exception

In-Cache Operation



(a)

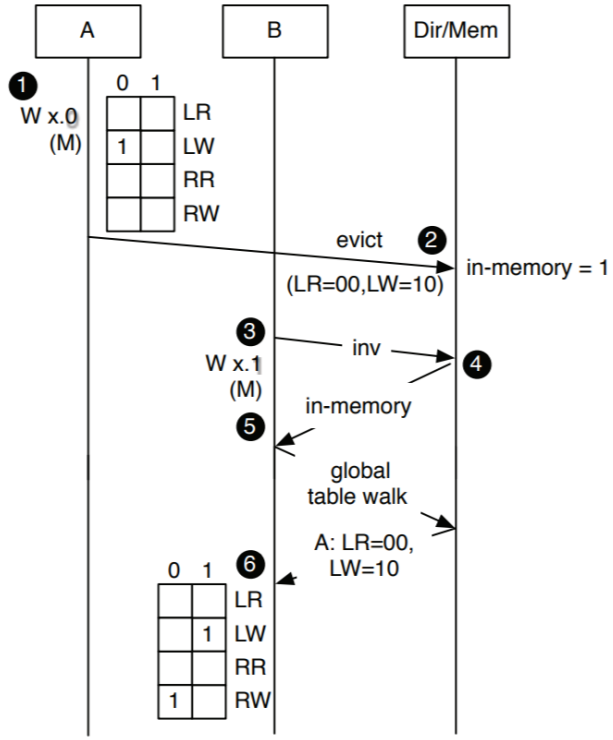


(b)

Electrical & Computer ENGINEERING

Architectural Support for Conflict Exception

Out-Cache Operation



Electrical & Computer ENGINEERING

Why Invariants hold?

#	State	Type of bit set	Reason why a set bit implies an access within an active region	Reason why an access within an active region implies a bit is set
1	Any	Local read bits	Follows directly from local access actions. Bits are set on access and cleared when a region ends.	
2		Local write bits		
3	M or E	Remote read bits	Remote bits are a combination of local bits from other threads. end-of-region message clears all remote bits previously set by data supplied to other caches, including the local cache.	Read misses can only result in exclusive state if no local read bits for the line are set for other threads. Transitions to M state require acknowledgments with local read and write bits from other threads.
4	O or S			Not enforced.
5	Any	Remote write bits		If an access happens after an in-region remote write, it will be satisfied with data supplied by the cache that wrote that byte or by a cache that wrote it later. The combination of local and remote write bits guarantees they are properly propagated.

Figure: Reasons why invariants hold

Electrical & Computer ENGINEERING

Results

App.	Regions		False Sharing / 1B Mem. Ops			EOR Messages			Mem. M-D Lkup / 100K Mem. Ops		Mem. Ovhd	Traffic Ovhd (B/MB)			
	Total #	Mem. Ops / Region	WaW	WaR	RaW	% Reg. w/ Msg.	Avg. # Lines	% to Mem.	Rem. Bits	Loc. Bits	(% Ftpt)	Rd Reply	Inv Ack	EOR Msgs	Evic.
blackscholes	50	12M	0.0	0.0	11.5	2.00	1	100	0.01	70.81	0.42	2.55	0.00	1.00	60K
bodytrack	37K	70K	8.0	3.4	15.6	0.02	3	100	0.01	58.41	2.29	0.63	0.46	0.87	31K
facesim	35K	2M	487.0	0.1	2251.5	0.05	297	100	0.14	285.09	8.58	62.49	13.52	3.60	58K
ferret	87K	30K	0.0	0.0	0.0	0.00	0	0	0.00	769.20	27.36	0.00	0.00	0.00	66K
fluidanimate	3M	996	0.0	0.0	1.4	0.00	1	100	0.01	9.60	1.58	0.23	0.00	0.13	30K
freqmine	96	54M	9.7	0.8	1048.0	5.21	112	100	0.05	213.29	17.09	63.89	0.64	11.47	64K
swaptions	96	20M	112.2	31.1	460.4	8.33	21	100	0.67	492.53	0.46	27.76	8.64	9.29	69K
vips	17K	252K	2.1	0.7	2.3	0.09	1	100	0.00	163.78	1.32	0.06	0.07	0.28	59K
x264	1K	1M	0.0	0.0	0.0	0.00	0	0	0.00	474.93	6.89	0.00	0.00	0.00	69K
canneal	118	39M	1.5	0.0	2.8	1.69	7	100	0.00	683.44	5.09	0.06	0.03	0.11	60K
dedup	18K	204K	68.8	28.7	96.5	1.92	1	100	0.07	104.96	1.01	3.63	3.67	11.85	49K
streamcluster	1K	1M	0.0	0.0	2.8	0.08	2	100	0.01	1465.45	7.10	0.03	0.00	0.02	53K
MySQL	1M	7K	0.0	0.0	0.1	0.00	1	100	0.00	24.15	0.24	0.01	0.00	0.02	50K
Apache	62K	27K	5.9	0.6	2.3	0.01	1	100	0.01	153.86	6.13	0.07	0.20	0.45	17K
Mean	304K	9M	49.7	4.7	278.2	1.39	32	86	0.07	354.96	6.11	11.53	1.95	2.79	52K

Figure: Overheads in detecting exceptions

Electrical & Computer ENGINEERING

Results reference

App.	KLOC	% Vio. Regions	# Vio PCs	# Vio Lines	# Vio Fns
blackscholes	0.5	0	0	0	0
bodytrack	5.9	5.6	161	22	16
facesim	22.6	6.8	101	21	13
ferret	9.2	1.4	1080	248	50
ferret†	9.2	0	0	0	0
fluidanimate	0.9	0	0	0	0
freqmine	2.7	0	0	0	0
swaptions	1.3	0	0	0	0
vips	109.3	2.6	234	133	41
x264	40.4	7.3	6	6	4
canneal	3.4	4.5	1	1	1
dedup	3.7	35.6	104	58	12
streamcluster	1.3	44.4	147	52	6
MySQL	1600.0	0.05	92	76	41
Apache	602.0	30.1	68	54	17
Mean	160.8	9.2	132.9	44.7	13.4

Figure: Number of exceptions in benchmarks used

Electrical & Computer ENGINEERING

DRFx

Motivation

- Full SC : Precludes common compiler and hardware optimizations, in order to ensure that the program semantics are respected. High performance overhead
- DRF models : Doesn't provide semantics for programs that contains data races
  - Racy execution can behave arbitrarily and violate desired safety properties
  - Debugging erroneous programs is difficult since, programmers has to always assume there may have been data races
- Debuggability
  - Writing race free programs is hard
  - Data races difficult to trigger, and even if triggered can manifest in many ways
  - Interaction of data races with compiler/hardware (which can re-order), are unknown to programmer
- It suffices to determine only data races that causes SC violations, and these can be determined in hardware using conflict detection

Electrical & Computer ENGINEERING

Contribution

- Halts program's execution on Dynamic **Memory Model**(MM) exception detection
- **DRF** : If a program is data-race free, then every execution of that program will be sequentially consistent and doesn't raise an MM exception
  - DRFx Compiler doesn't introduce additional data races - eg., speculative reads and writes and disables
  - Programs is split into synchronization free regions and optimizations allowed only within those regions, thus can't be reordered in a manner that violates SC
- **Soundness** : If SC is violated, program terminates with MM exception
  - Sufficient to determine region conflict between concurrent regions
  - Hardware based data race detection - software based slows down the execution of program by 8 times
- **Safety** : If an execution invokes system call, the . observable kernel state is reachable through SC
  - System calls to be places within in its own region

Electrical & Computer ENGINEERING

Compiler Design

- Hard Fences - DRF and safety
  - Compiler must insert hard fence before and after each synchronization access
  - Hard fence for system call invocation, before entering and after exiting kernel
  - LLVM fence - doesn't allow memory reordering across fence
- Soft Fences - Boundedness (Finite hardware resources)
  - Statically bound the number of memory operations in a region
  - Before each function call and return , before each loop back-edge, and extra to bound region sizes
  - The max. number of memory operations = size of hardware buffer. Exposing hardware information
  - Prevents compiler optimizations across soft fences, eg., across loop iterations.
- Compiler Optimizations
  - Instead of speculative read, specific prefetch instruction which hardware wouldn't track for conflict detection
  - Loop invariant code motion is allowed, as long as loop body guarantees atomicity

Electrical & Computer ENGINEERING

Hardware design

- Region Buffer
  - Circular queue buffer, storing region start/end timestamps, and read/write operations, with bits indicating the number of bytes per access (Allows granularity)
- Lazy Conflict detection
  - When all of regions's instructions have completed, processor broadcasts region's read/write to all other processors (High-traffic)
  - Non-SC state for sometime before violation is detected
    - Execution causes exception apart from MM exception - additional MM exception check has to be done at this point
    - Execution enters a non-terminating loop - Conflict detection if a region has executed for more than C cycles
- Handling Hard fences
  - Processor stalls until all memory executions and coherence as part of region completes, then initiate conflict detection (Performance overhead)
- Handling Soft fences
  - Can safely reorder memory operations across regions delimited by soft fences
  - Can run out of region buffer case (since bounded), in that case core has to be stalled till next commit

Electrical & Computer ENGINEERING

Results

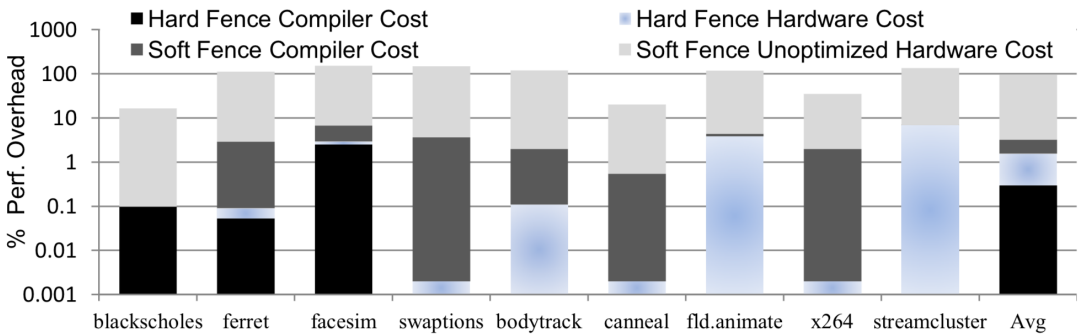


Figure 5. Performance overhead with respect to the performance of an uninstrumented binary, compiled with all the traditional compiler optimizations enabled.

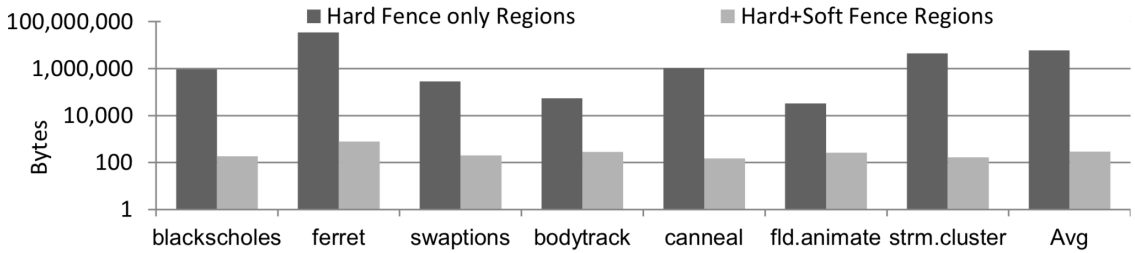


Figure 6. Maximum number of unique memory bytes accessed in any region.

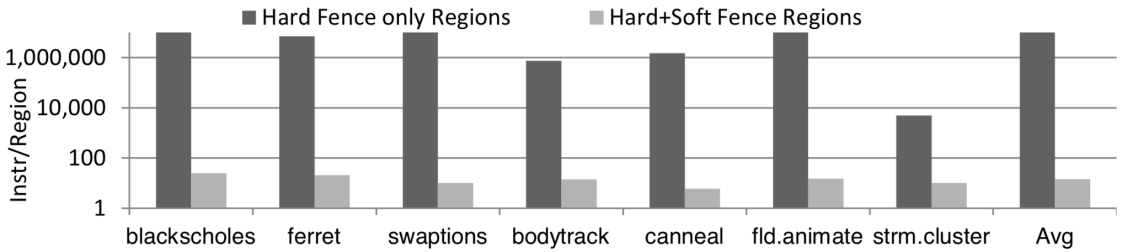


Figure 7. Average number of instructions executed in a region.

cal & Computer ENGINEERING