Evaluating the Performance of Atomic Operations on Modern Multicore Systems

Goncharenko E. A., Paznikov A. A., Tabakov A.V.

kesvesna@rambler.ru, apaznikov@gmail.com, komdosh@yandex.ru

Conference on Applied Physics, Information Technologies and Engineering» (APITECH-2019)

Saint Petersburg Electrotechnical University “LETI”

Atomic operations

- The operation is atomic, if it's performed in one undividable step relative to other threads. In other words, no thread can observe this operation as "partially completed".
- If two or more threads perform operations with a shared variable and at least one of them writes (stores) to it, then the threads must use atomic operations to avoid data races.
- These operations implemented in hardware as processor instructions are highly demanded in multithreaded programming

Execution time of atomic operations depends on:

- Cache coherence protocol (the state of cache-line)
- Data locality (local, remote cache, RAM, NUMA)
- Buffer size
- Processor microarchitecture

We analyze the efficiency of atomic operations (Load, Store, FAA, SWP, CAS) considering buffer sizes, data locality and cache-line states.

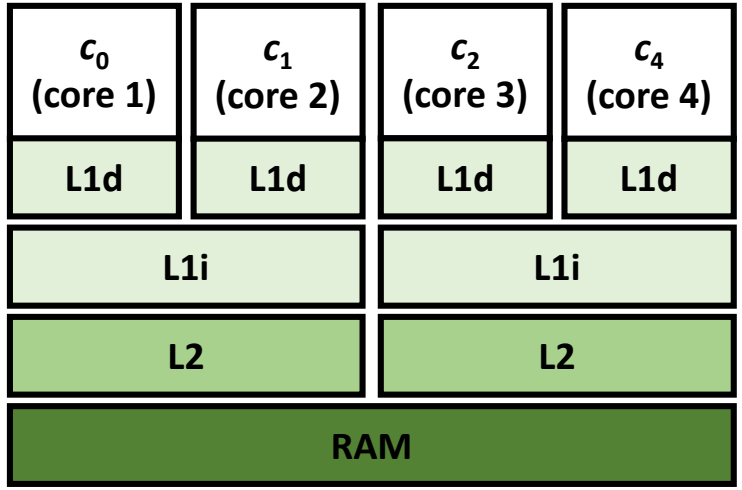
Algorithms of experiments

- Perform atomic operations with the data in cache-memory.
- Compute execution time of one atomic operation (latency), evaluate throughput

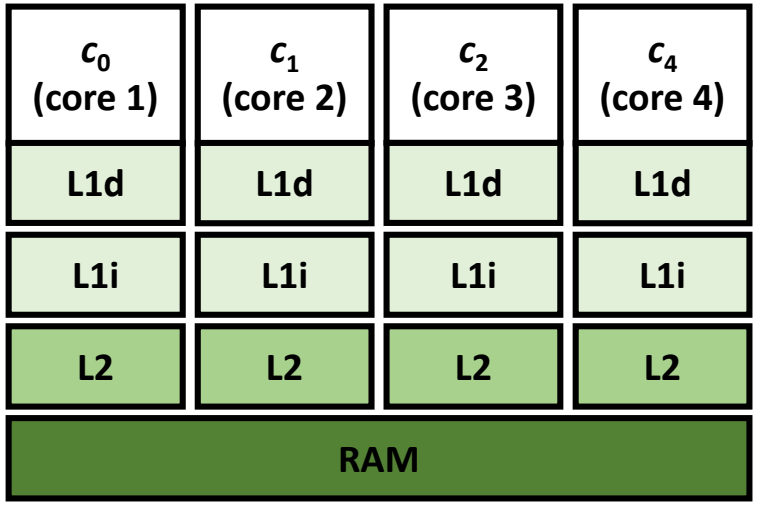
```
for k = 0 to z do                          ▷ iterate through the buffer
SetAffinity(c[i])                          ▷ choose the core to the test
SetState(d[i])                             ▷ set the state of cache-line (in coherence protocol)
        for j = 0 to m do                  ▷ number of iterations
                t1 = GetTime()             ▷ begin time
                for i = 0 to d do          ▷ size of the data copied in cache
                        AtomicOp(q[i])     ▷ perform atomic operation
                end for
                t2 = GetTime()             ▷ end time
        end for
        t = (t2 - t1) / m                  ▷ result
end for
```

Test architectures

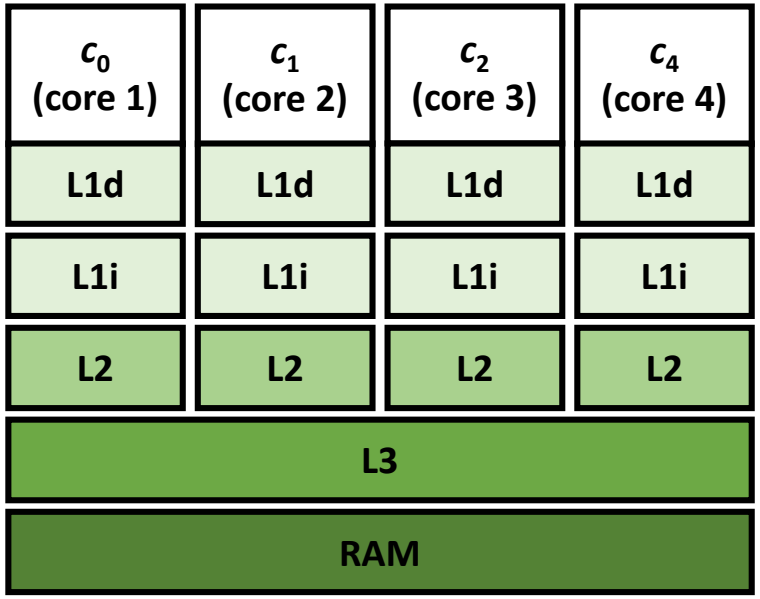
AMD A10 - 4600M (Piledriver)



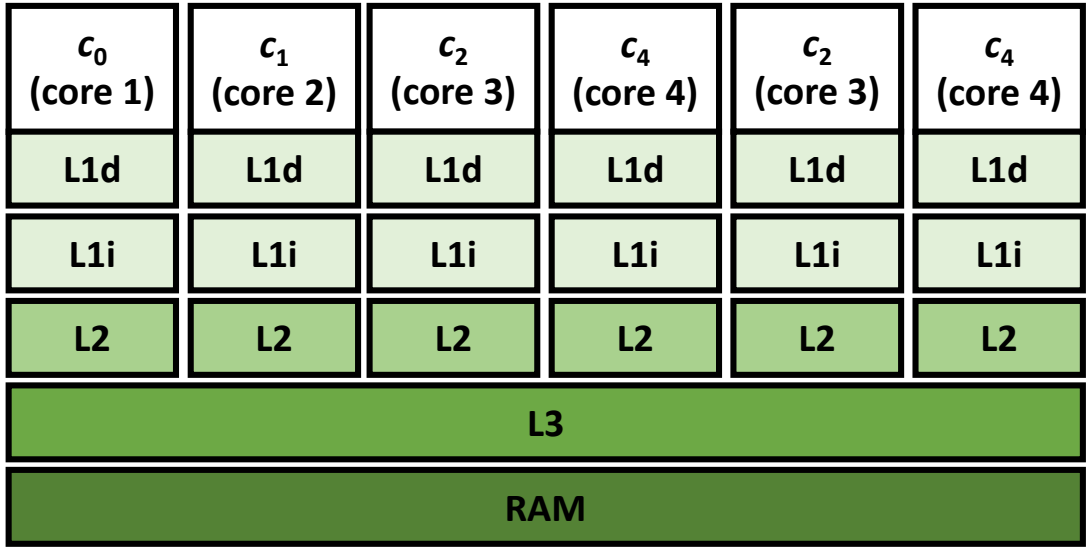
AMD Athlon 2 X4 640 (K10)



Intel Xeon E5540 (Nehalem-EP)



Intel Xeon X5670 (Westmere-EP)



Latency for Westmere-EP, SWP operation

	c_0					c_1					c_2			
	L1	L2	L3	RAM		L1	L2	L3	RAM		L1	L2	L3	RAM
E	1,89	1,89	1,89	1,89-1,91	E	1,92-1,95	1,95-1,97	1,97-1,93	1,93-1,94	E	1,92-1,95	1,95-1,97	1,97-1,93	1,93-1,94
I	1,93	1,91	1,91	1,93	I	1,94	1,94-1,96	1,96-1,93	1,93	I	1,94	1,94-1,96	1,96-1,93	1,93
M	1,94-1,89	1,89-1,94	1,94-1,95	1,93	M	1,94	1,94	1,94-1,93	1,93	M	1,94	1,94	1,94-1,93	1,93
S	2,01	2,01	2,01-2	2,00	S	2,02	2,07	2,07-2	2,00	S	2,02	2,07	2,07-2	2,00

- minimal latency, ns

- maximum latency, ns

c_0, c_1, c_2 – cores 1, 2, 3

E, I, M, S – cache-line states

RAM – data size exceeds cache-memory

L1, **L2**, **L3** – data size exceeds doesn't exceeds cache-memory of 1st, 2nd and 3rd level

Conclusion

- We experimentally show that operations **“unsuccessful CAS”, FAA** and **SWP** has the minimum latency.
- **Load** operation has the minimum latency
- **“successful CAS”** and **store** have maximum latency.
- We analyzed the experimental results and gave the recommendations for increasing the throughput and minimizing the latency of atomic operation performance of modern processors.
- So, the application of our recommendations will increase the throughput of atomic operations on the Piledriver processors from 1.1 to 3.9 times, on the K10 processor - from 1.1 to 1.6 times, on the Nehalem-EP processor from 2.1 to 6, 1 time, on the Westmere-EP processor - from 1.1 to 7.2 times.
- Thus, the results show that the execution time of atomic operations can vary widely, depending on the conditions of their execution (cache line state, localization, and buffer size). These evidences should be considered for designing new concurrent data structures and synchronization primitives.