Algorithms for Optimization of Remote Core Locking Synchronization in Hierarchical Multicore Computer Systems

Alexey Paznikov
Saint Petersburg Electrotechnical University "LETI"

apaznikov@gmail.com

Abstract. This paper proposes the algorithms for optimization of Remote Core Locking (RCL) synchronization method in multithreaded programs. The algorithm of initialization of RCL-locks and the algorithms for threads affinity optimization are developed. The algorithms consider the structures of hierarchical computer systems and non-uniform memory access (NUMA) to minimize execution time of RCL-programs.

Keywords: remote core locking, RCL, synchronization, critical sections, scalability

Full paper: https://www.dropbox.com/s/rp89f60hozdoljt/paznikov-rcl.pdf?dl=0

This paper considers Remote Core Locking (RCL) synchronization in multithreaded programs, which assumes execution of critical section on the dedicated processor cores. The current implementation of RCL has several drawbacks. There are no memory affinity in NUMA systems. RCL also has no mechanism of automatic selection of processor cores for server thread and working threads.

For the memory affinity optimization and RCL-server processor affinity optimization we propose the algorithm RCLLockInitNUMA of initialization of RCL-lock. On the first stage we compute the number of processor cores which is not busy by the RCL-server and the number of processor cores used on each of NUMA-node. The second stage of the algorithm includes the search of suboptimal processor core and the affinity of RCL-server to it. The algorithms is finished by call of function of RCL-lock initialization with selected affinity.

For the optimization of working thread affinity we propose the heuristic algorithm RCLHierarchicalAffinity. The algorithm takes into account hierarchical structure of multi-core CS to minimize the execution time of multithreaded programs with RCL. That algorithm is executed each time when parallel thread is created.

The figure 1 depicts the throughput b of critical section with number p of threads. The algorithm RCLLockInitNUMA minimizes by $10-20\%$ the throughput of critical section at random access to the elements of test array and at

strided access. The figures 2 represent the experimental results of different affinities for the benchmark. The algorithm RCLHierarchicalAffinity significantly increases critical section throughput.

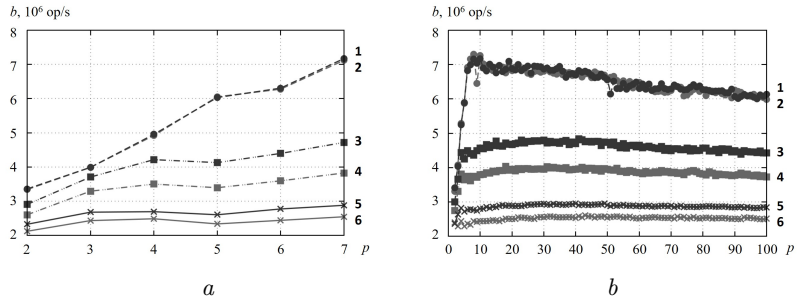

Fig. 1: Efficiency of the algorithms of RCL-lock initalization. a – $p = 2, \ldots, 7$, b – $p = 2, \ldots, 100$ 1 – RCLLockInitNUMA, sequential access, 2 – RCLLockInitDefault, sequential access, 3 – RCLLockInitNUMA, strided access, 4 – RCLLockInitDefault, strided access, 5 – RCLLockInitNUMA, random access, 6 – RCLLockInitDefault, random access

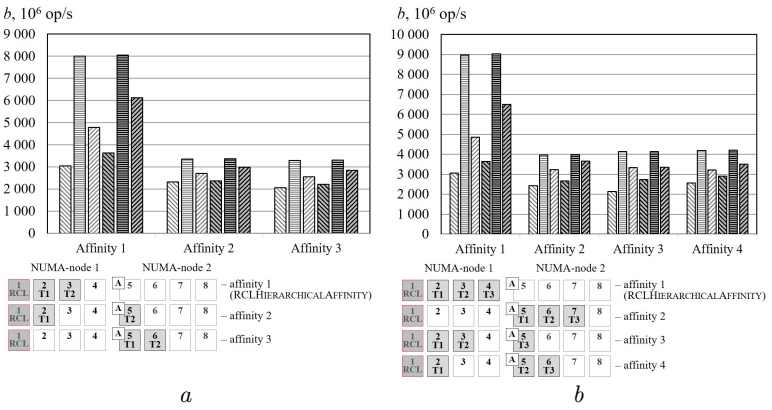

Fig. 2: Threads affinity efficiency comparison. a – $p = 2$, b – $p = 3$, c – $p = 4$, d – $p = 5$ ▧ – RCLLockInitDefault, random access, ▤ – RCLLockInitDefault, sequential access, ▨ – RCLLockInitDefault, strided access, ▧ – RCLLockInitNUMA, random access, ▤ – RCLLockInitNUMA, sequential access, ▨ – RCLLockInitNUMA, strided access. 2 T1 – working thread, 1 RCL – RCL-server, A – thread allocating the memory.

The algorithm RCLLockInitNUMA increases by $10 - 20\%$ at the average the throughput of critical sections on the NUMA systems. The optimization is reached by means of minimization of number of addresses to remote memory NUMA-segments. The algorithm RCLHierarchicalAffinity increases the throughput of critical section up to $1.2 - 2.4$ times for all access templates on some computer systems. The algorithms sets the affinity with considering the hierarchical levels of multicore computer systems.