

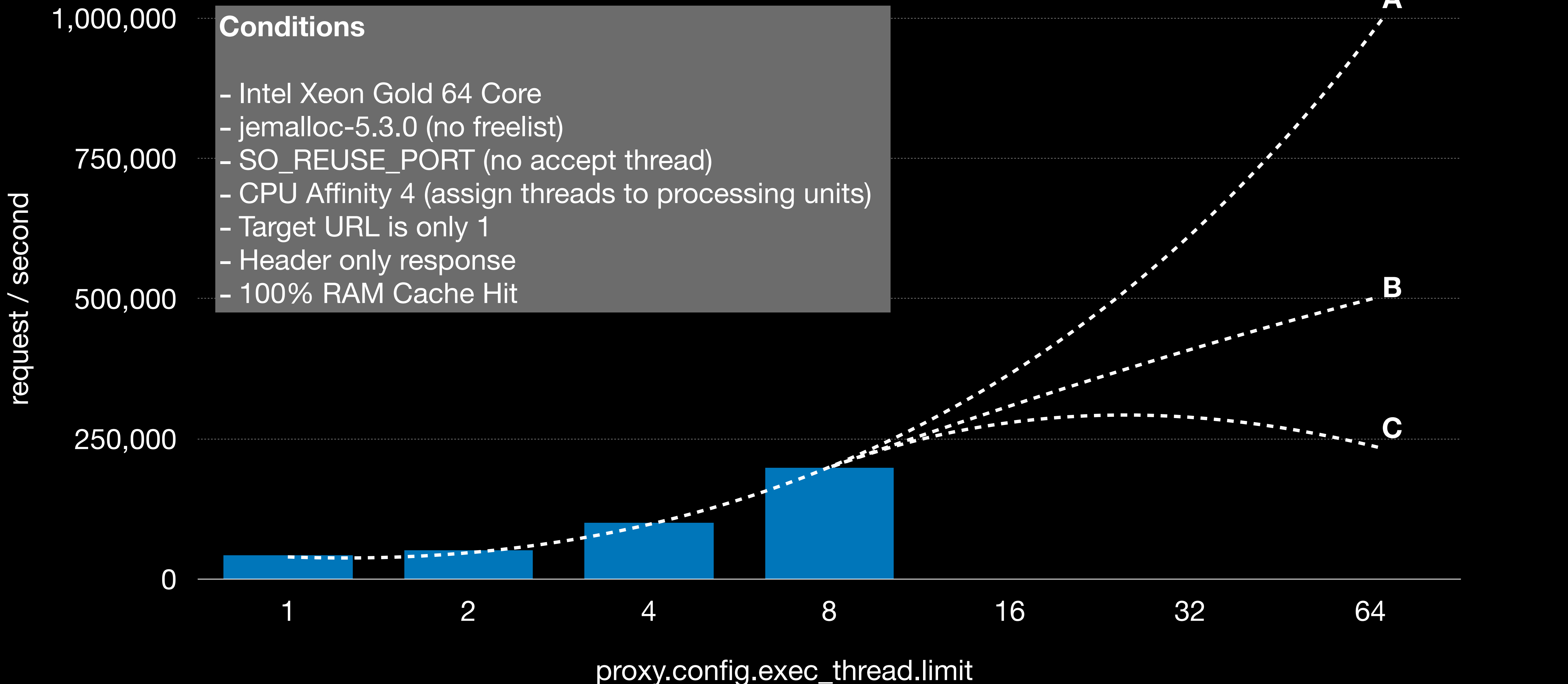
Accelerating Traffic Server

ATS Spring 2023 Summit

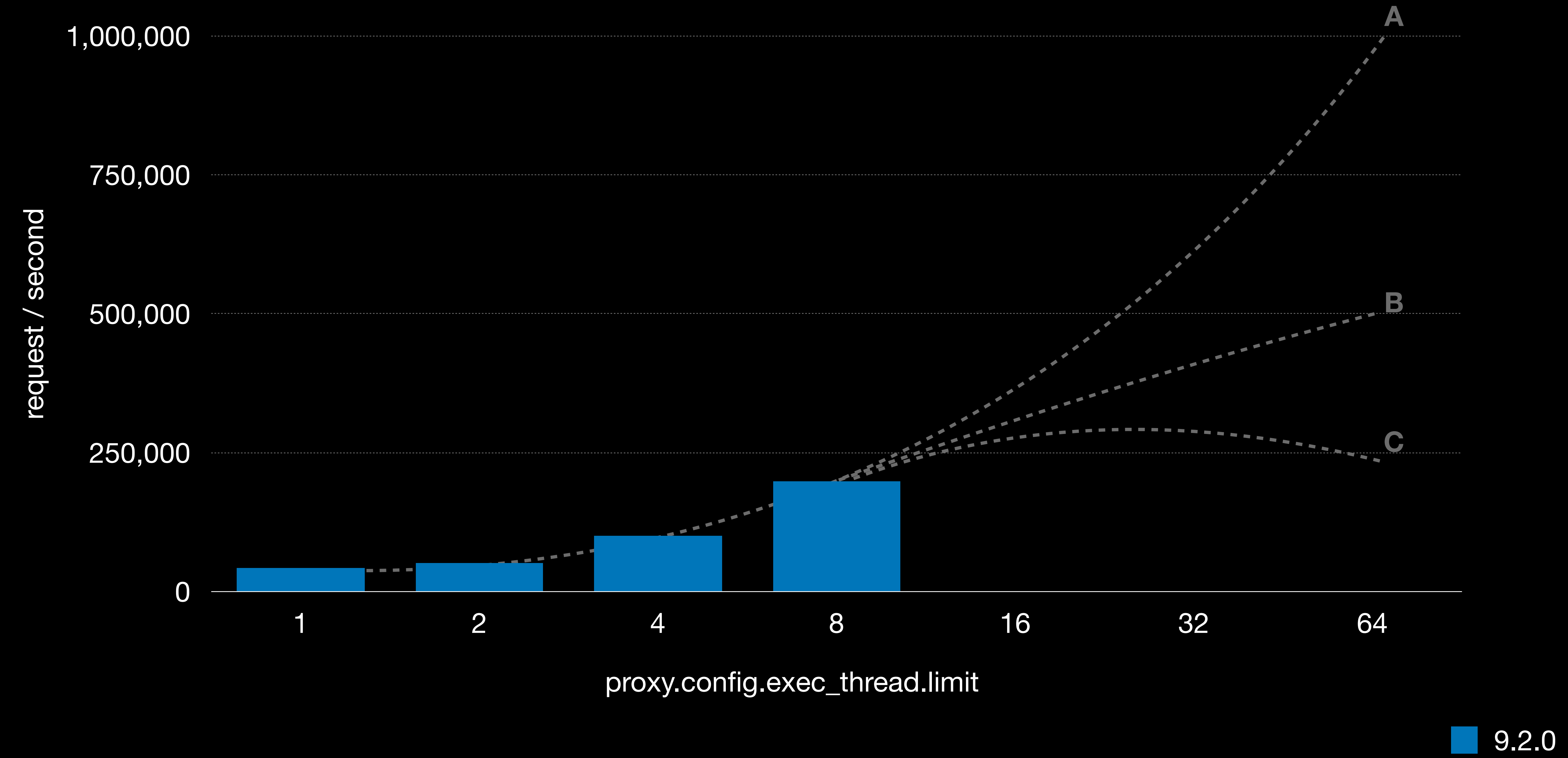
Masaori Koshiba (masaori@apache.org)

Quiz

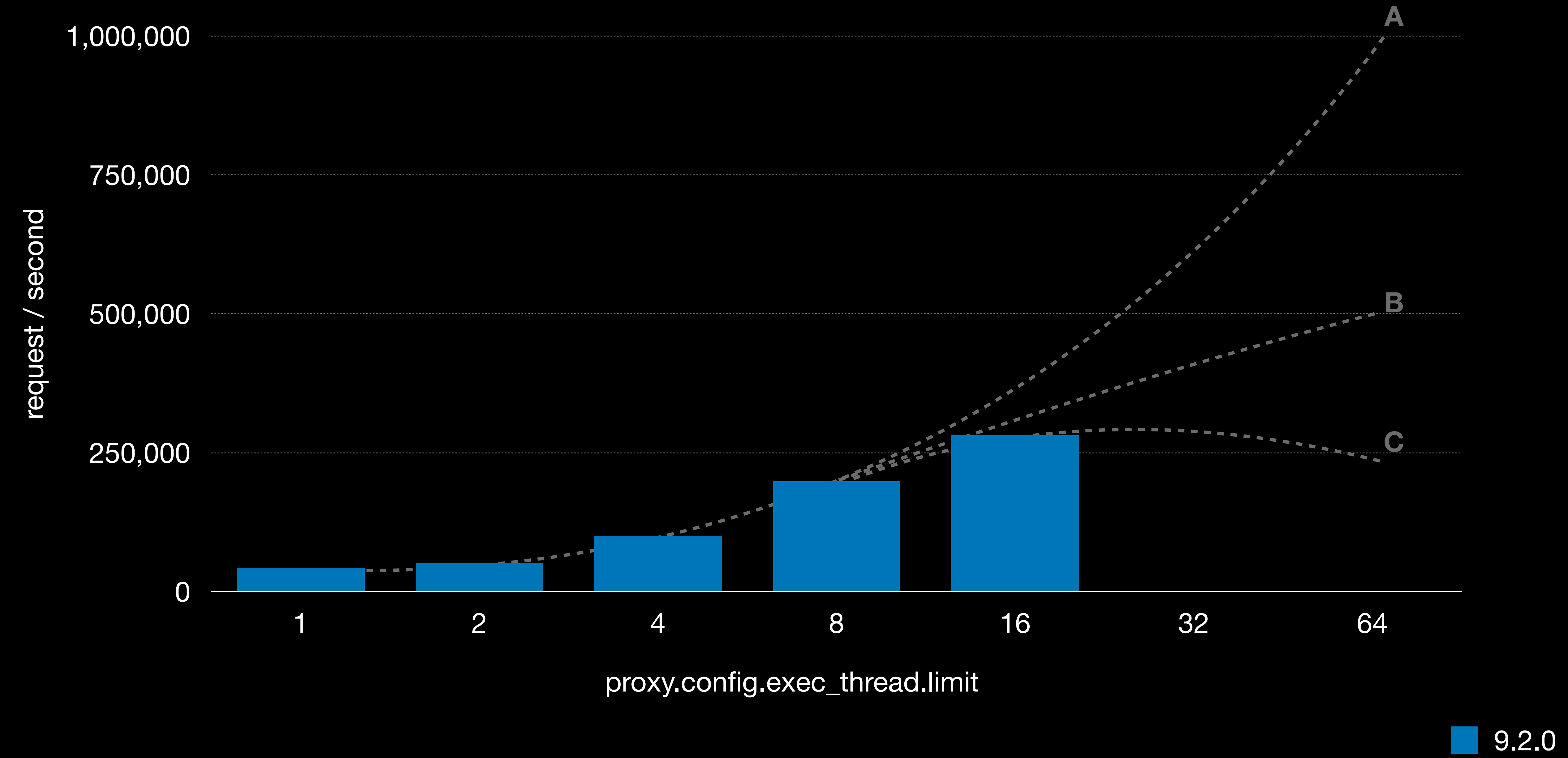
Q: Does ATS scale with the number of cores?



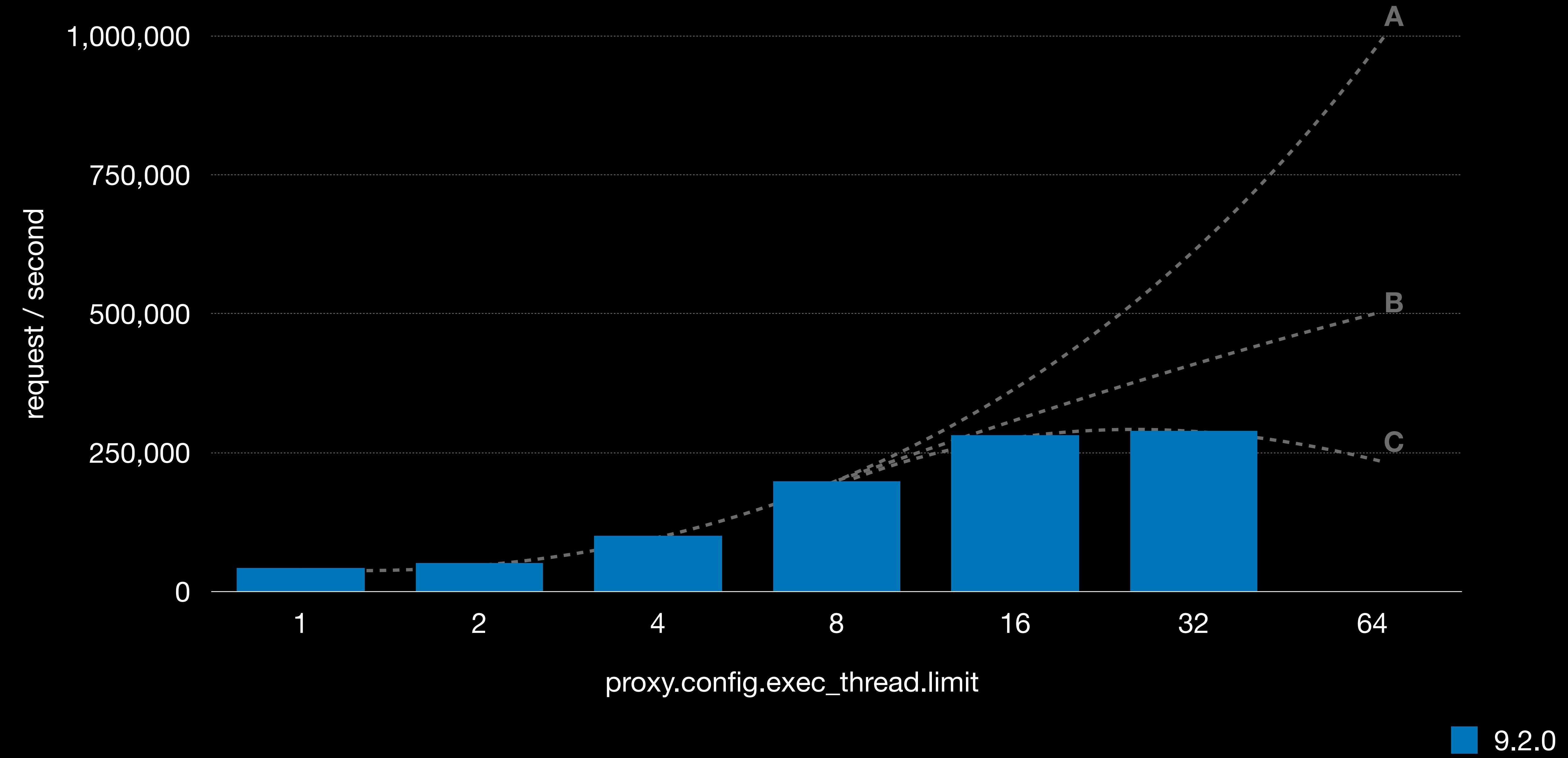
Q: Does *ATS* scale with the number of cores?



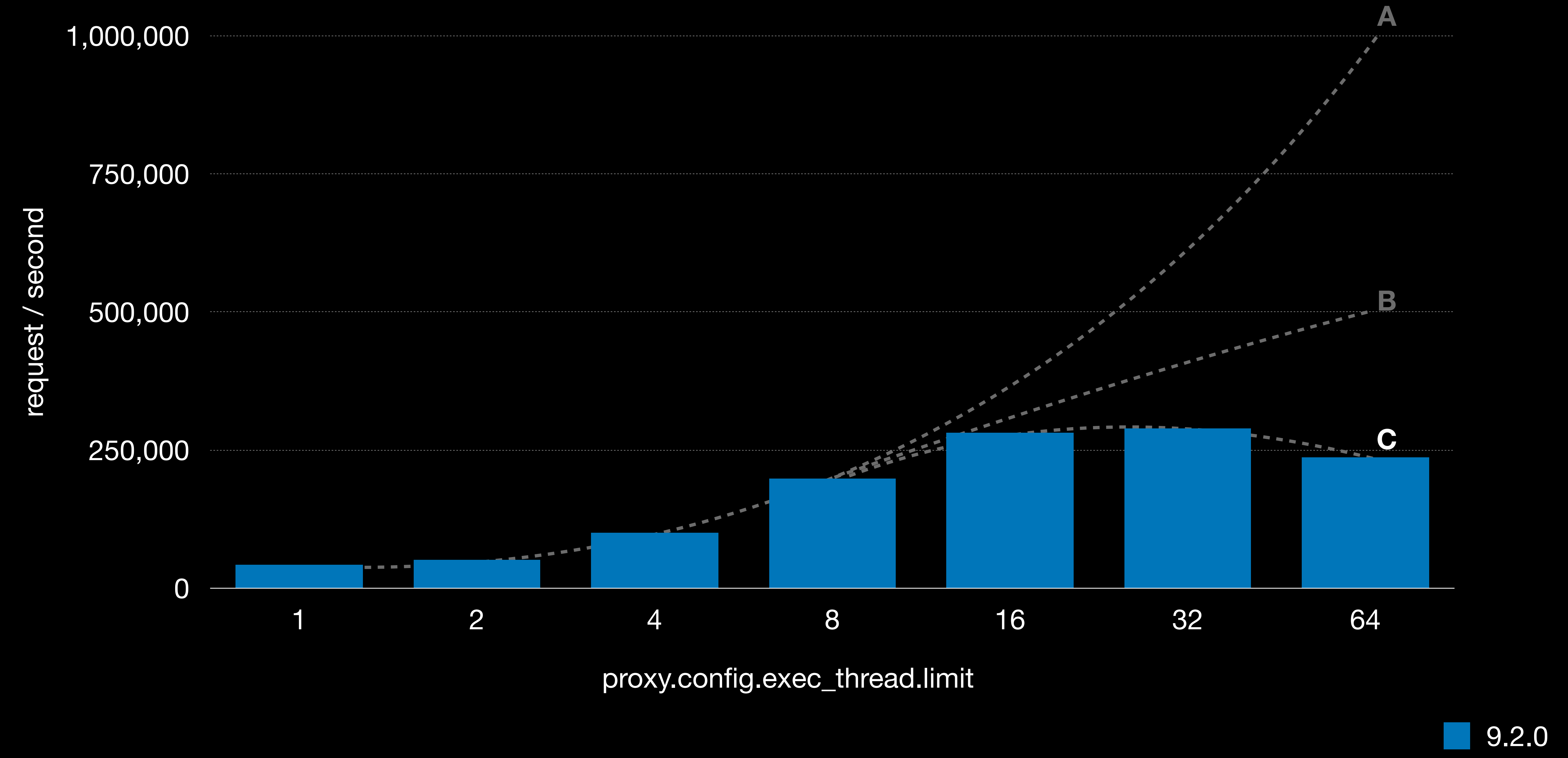
Q: Does *ATS* scale with the number of cores?



Q: Does *ATS* scale with the number of cores?



Q: Does *ATS* scale with the number of cores?



1 thread

```
1 [|||||100.0%] 17 [ 0.0%] 33 [|| 2.6%] 49 [ 0.0%]
2 [ 0.0%] 18 [ 0.7%] 34 [ 0.0%] 50 [ 0.0%]
3 [ 0.0%] 19 [|| 1.3%] 35 [ 0.0%] 51 [ 0.0%]
4 [ 0.0%] 20 [ 0.0%] 36 [ 0.0%] 52 [ 0.0%]
5 [ 0.0%] 21 [ 0.6%] 37 [ 0.0%] 53 [ 0.0%]
6 [ 0.0%] 22 [ 0.0%] 38 [ 0.0%] 54 [ 0.0%]
7 [ 0.0%] 23 [ 0.0%] 39 [ 0.0%] 55 [ 0.0%]
8 [ 0.0%] 24 [ 0.6%] 40 [ 0.0%] 56 [ 0.0%]
9 [|| 1.9%] 25 [ 0.0%] 41 [ 0.0%] 57 [ 0.0%]
10 [ 0.0%] 26 [ 0.0%] 42 [ 0.0%] 58 [ 0.0%]
11 [ 0.0%] 27 [ 0.0%] 43 [ 0.0%] 59 [ 0.0%]
12 [ 0.0%] 28 [ 0.0%] 44 [ 0.0%] 60 [ 0.0%]
13 [ 0.0%] 29 [ 0.0%] 45 [ 0.0%] 61 [ 0.0%]
14 [ 0.0%] 30 [ 0.0%] 46 [ 0.0%] 62 [ 0.0%]
15 [ 0.0%] 31 [ 0.7%] 47 [ 0.0%] 63 [ 0.6%]
16 [ 0.7%] 32 [ 0.0%] 48 [ 0.0%] 64 [ 0.0%]
Mem[||||| 7.10G/376G] Tasks: 120, 513 thr; 2 running
Swp[ 0K/23.4G] Load average: 0.38 0.34 0.28
Uptime: 62 days, 21:09:36
```

4 threads

```
1 [|||||100.0%] 17 [ 0.7%] 33 [|||||100.0%] 49 [ 0.0%]
2 [|||||100.0%] 18 [ 0.7%] 34 [|||||100.0%] 50 [ 0.7%]
3 [ 0.0%] 19 [ 0.0%] 35 [ 0.7%] 51 [ 0.0%]
4 [ 0.0%] 20 [ 0.7%] 36 [ 0.0%] 52 [ 0.0%]
5 [ 0.0%] 21 [ 0.0%] 37 [ 0.0%] 53 [ 0.7%]
6 [ 0.0%] 22 [|| 1.3%] 38 [ 0.7%] 54 [|| 4.1%]
7 [ 0.0%] 23 [ 0.0%] 39 [ 0.0%] 55 [ 0.7%]
8 [ 0.0%] 24 [ 0.0%] 40 [ 2.0%] 56 [ 0.0%]
9 [ 0.0%] 25 [ 0.0%] 41 [ 0.0%] 57 [ 0.0%]
10 [ 0.0%] 26 [ 0.7%] 42 [ 2.0%] 58 [ 0.0%]
11 [ 0.0%] 27 [|| 2.0%] 43 [ 0.0%] 59 [ 0.0%]
12 [ 0.0%] 28 [ 0.0%] 44 [ 0.0%] 60 [ 0.7%]
13 [ 0.0%] 29 [ 0.0%] 45 [ 0.0%] 61 [ 0.0%]
14 [ 0.0%] 30 [ 0.0%] 46 [ 0.0%] 62 [ 0.0%]
15 [ 0.0%] 31 [ 0.0%] 47 [ 0.0%] 63 [ 0.0%]
16 [ 0.0%] 32 [ 0.0%] 48 [ 0.7%] 64 [ 1.3%]
Mem[||||| 7.23G/376G] Tasks: 123, 517 thr; 5 running
Swp[ 0K/23.4G] Load average: 1.18 0.57 0.32
Uptime: 63 days, 21:14:37
```

2 threads

```
1 [|||||100.0%] 17 [ 0.0%] 33 [|||||100.0%] 49 [|| 2.6%]
2 [ 0.0%] 18 [ 0.0%] 34 [ 0.0%] 50 [|| 1.3%]
3 [ 0.0%] 19 [ 0.7%] 35 [ 0.0%] 51 [ 0.0%]
4 [ 0.0%] 20 [ 0.0%] 36 [ 0.0%] 52 [ 0.0%]
5 [ 0.0%] 21 [ 2.6%] 37 [ 0.0%] 53 [ 0.0%]
6 [ 0.0%] 22 [ 0.7%] 38 [|| 4.0%] 54 [ 0.0%]
7 [ 0.0%] 23 [ 0.0%] 39 [ 0.7%] 55 [ 0.0%]
8 [ 0.0%] 24 [ 0.7%] 40 [ 0.0%] 56 [ 0.0%]
9 [ 0.0%] 25 [ 0.0%] 41 [ 0.0%] 57 [ 0.0%]
10 [ 0.0%] 26 [ 0.0%] 42 [ 0.0%] 58 [ 0.0%]
11 [ 0.0%] 27 [ 0.7%] 43 [|| 4.0%] 59 [ 0.0%]
12 [ 0.0%] 28 [ 0.7%] 44 [ 0.0%] 60 [ 0.0%]
13 [ 0.0%] 29 [ 0.0%] 45 [ 0.0%] 61 [ 0.0%]
14 [ 0.0%] 30 [ 0.0%] 46 [ 0.0%] 62 [ 0.0%]
15 [ 0.0%] 31 [ 0.0%] 47 [ 0.7%] 63 [ 0.0%]
16 [ 0.0%] 32 [ 0.0%] 48 [ 0.7%] 64 [ 0.0%]
Mem[||||| 7.13G/376G] Tasks: 119, 515 thr; 3 running
Swp[ 0K/23.4G] Load average: 0.40 1.22 1.33
Uptime: 62 days, 21:26:00
```

8 threads

```
1 [|||||100.0%] 17 [ 0.0%] 33 [|||||100.0%] 49 [ 0.7%]
2 [|||||100.0%] 18 [ 0.0%] 34 [|||||100.0%] 50 [ 0.0%]
3 [|||||100.0%] 19 [ 0.7%] 35 [|||||100.0%] 51 [ 0.0%]
4 [|||||100.0%] 20 [ 3.0%] 36 [|||||100.0%] 52 [ 0.0%]
5 [ 0.0%] 21 [ 0.0%] 37 [ 0.7%] 53 [|| 4.7%]
6 [ 0.0%] 22 [|| 1.4%] 38 [ 0.0%] 54 [|| 8.5%]
7 [ 0.0%] 23 [|| 2.0%] 39 [ 0.0%] 55 [ 0.0%]
8 [ 0.0%] 24 [|| 3.5%] 40 [ 0.0%] 56 [ 0.0%]
9 [ 0.0%] 25 [ 0.0%] 41 [ 0.7%] 57 [ 1.4%]
10 [ 0.0%] 26 [ 0.0%] 42 [ 1.4%] 58 [ 2.8%]
11 [ 0.0%] 27 [ 0.7%] 43 [ 0.0%] 59 [ 1.5%]
12 [ 0.0%] 28 [ 1.4%] 44 [ 0.0%] 60 [ 0.0%]
13 [ 0.0%] 29 [ 0.7%] 45 [ 0.7%] 61 [ 0.0%]
14 [ 0.0%] 30 [ 0.7%] 46 [ 0.7%] 62 [ 3.3%]
15 [ 0.0%] 31 [ 0.7%] 47 [ 2.0%] 63 [ 0.0%]
16 [ 0.0%] 32 [ 0.0%] 48 [ 1.4%] 64 [ 0.0%]
Mem[||||| 7.12G/376G] Tasks: 119, 520 thr; 9 running
Swp[ 0K/23.4G] Load average: 2.50 2.15 1.13
Uptime: 62 days, 21:16:23
```


16 threads

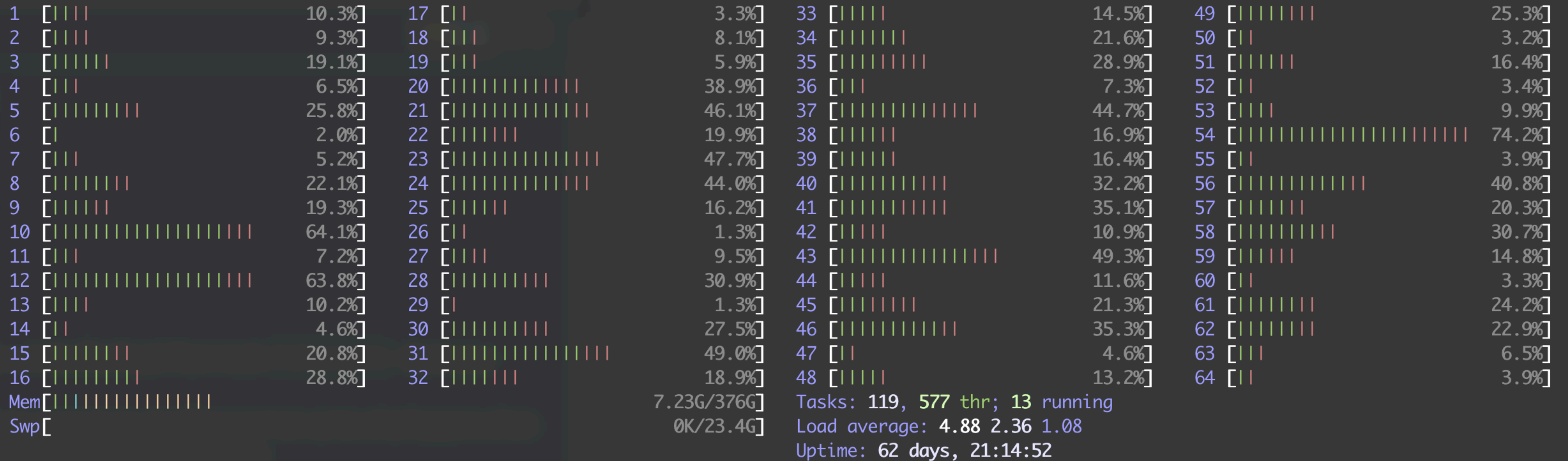
```
1 [|||||||||||||||||100.0%] 17 [ 0.0%] 33 [|||||||||||||||||100.0%] 49 [ 1.4%]
2 [|||||||||||||||||100.0%] 18 [ 0.0%] 34 [|||||||||||||||||100.0%] 50 [ 0.0%]
3 [|||||||||||||||||100.0%] 19 [ 0.0%] 35 [|||||||||||||||||100.0%] 51 [ 0.0%]
4 [|||||||||||||||||100.0%] 20 [ 3.0%] 36 [|||||||||||||||||100.0%] 52 [ 0.7%]
5 [|||||||||||||||||100.0%] 21 [ 5.3%] 37 [|||||||||||||||||100.0%] 53 [ 1.4%]
6 [|||||||||||||||||100.0%] 22 [ 0.7%] 38 [|||||||||||||||||100.0%] 54 [ 3.9%]
7 [|||||||||||||||||100.0%] 23 [ 5.9%] 39 [|||||||||||||||||100.0%] 55 [ 0.0%]
8 [|||||||||||||||||100.0%] 24 [ 6.5%] 40 [|||||||||||||||||99.4%] 56 [ 0.7%]
9 [ 4.2%] 25 [ 13.1%] 41 [ 0.7%] 57 [ 2.2%]
10 [ 0.0%] 26 [ 0.7%] 42 [ 2.8%] 58 [ 0.7%]
11 [ 0.0%] 27 [ 4.8%] 43 [ 0.7%] 59 [ 2.2%]
12 [ 0.0%] 28 [ 1.4%] 44 [ 0.7%] 60 [ 0.0%]
13 [ 0.7%] 29 [ 1.4%] 45 [ 2.9%] 61 [ 0.7%]
14 [ 0.7%] 30 [ 0.7%] 46 [ 0.0%] 62 [ 0.0%]
15 [ 0.0%] 31 [ 0.0%] 47 [ 0.0%] 63 [ 0.0%]
16 [ 0.0%] 32 [ 10.7%] 48 [ 0.0%] 64 [ 1.3%]
Mem[|||||] 7.14G/376G Tasks: 119, 528 thr; 17 running
Swp[ ] 0K/23.4G Load average: 2.74 2.19 1.18
Uptime: 62 days, 21:17:09
```

32 threads

```
1 [||||| 40.8%] 17 [ 3.4%] 33 [||||| 56.6%] 49 [ 0.0%]
2 [||||| 30.8%] 18 [ 6.3%] 34 [||||| 74.3%] 50 [ 2.0%]
3 [||||| 57.2%] 19 [ 11.4%] 35 [||||| 76.8%] 51 [ 1.4%]
4 [||||| 12.5%] 20 [ 8.4%] 36 [||||| 78.9%] 52 [ 2.8%]
5 [||||| 84.9%] 21 [ 1.3%] 37 [||||| 49.3%] 53 [ 2.0%]
6 [||||| 28.9%] 22 [ 0.0%] 38 [||||| 43.0%] 54 [ 4.6%]
7 [||||| 23.7%] 23 [ 0.0%] 39 [||||| 59.9%] 55 [ 0.0%]
8 [||||| 31.5%] 24 [ 0.7%] 40 [||||| 34.2%] 56 [ 0.7%]
9 [||||| 92.2%] 25 [ 6.2%] 41 [||||| 36.1%] 57 [ 4.2%]
10 [||||| 42.4%] 26 [ 6.4%] 42 [||||| 61.2%] 58 [ 6.2%]
11 [||||| 54.9%] 27 [ 2.9%] 43 [||||| 30.2%] 59 [ 2.1%]
12 [||||| 80.9%] 28 [ 1.4%] 44 [||||| 12.4%] 60 [ 1.3%]
13 [||||| 49.3%] 29 [ 2.7%] 45 [||||| 60.9%] 61 [ 2.0%]
14 [||||| 46.8%] 30 [ 6.2%] 46 [||||| 55.6%] 62 [ 1.3%]
15 [||||| 40.3%] 31 [ 0.7%] 47 [||||| 23.2%] 63 [ 0.0%]
16 [||||| 74.3%] 32 [ 8.2%] 48 [||||| 47.7%] 64 [ 1.4%]
Mem[|||||] 7.18G/376G Tasks: 116, 544 thr; 18 running
Swp[ ] 0K/23.4G Load average: 5.71 3.22 1.65
Uptime: 62 days, 21:18:56
```

"Underutilized" Issue

64 threads



Profiling ATS

Cache Enabled (almost 100% Hit)

thread 38875

mutex [unknown] ::: wait time 0.00us ::: hold time 24692.07us ::: enter count 3051 ::: try-lock failure count 13693

CacheVC::openReadStartHead(int, Event*)+0xd8 [traffic_server] (55e891616368)

EThread::process_event(Event*, int)+0x276 [traffic_server] (55e89170b586)

EThread::execute_regular()+0x33d [traffic_server] (55e89170bf0d)

EThread::execute()+0x171 [traffic_server] (55e89170c361)

spawn_thread_internal(void*)+0x55 [traffic_server] (55e89170a8a5)

start_thread+0xc5 [libpthread-2.17.so] (7f1216cf8ea5)

mutex [unknown] ::: wait time 0.00us ::: hold time 10263.61us ::: enter count 1372 ::: try-lock failure count 6294

CacheVC::openReadClose(int, Event*)+0xab [traffic_server] (55e89161a1db)

EThread::process_event(Event*, int)+0x276 [traffic_server] (55e89170b586)

EThread::execute_regular()+0x33d [traffic_server] (55e89170bf0d)

EThread::execute()+0x171 [traffic_server] (55e89170c361)

spawn_thread_internal(void*)+0x55 [traffic_server] (55e89170a8a5)

start_thread+0xc5 [libpthread-2.17.so] (7f1216cf8ea5)

mutex [unknown] ::: wait time 0.00us ::: hold time 3041.71us ::: enter count 337 ::: try-lock failure count 1391

Cache::open_read(Continuation*, ats::CryptoHash const*, HTTPHdr*, OverridableHttpConfigParams const*, CacheFra

CacheProcessor::open_read(Continuation*, HttpCacheKey const*, HTTPHdr*, OverridableHttpConfigParams const*, lo

HttpCacheSM::open_read(HttpCacheKey const*, URL*, HTTPHdr*, OverridableHttpConfigParams const*, long)+0x9d [tr

HttpSM::do_cache_lookup_and_read()+0x169 [traffic_server] (55e89147d929)

HttpSM::set_next_state()+0x4d1 [traffic_server] (55e8914820f1)

HttpSM::set_next_state()+0xb48 [traffic_server] (55e891482768)

HttpSM::set_next_state()+0xa06 [traffic_server] (55e891482626)

Profiling ATS

Cache Enabled (almost 100% Hit)

thread 38875

```
mutex [unknown] ::: wait time 0.00us ::: hold time 24692.07us ::: enter count 3051 ::: try-lock failure count 13693
  CacheVC::openReadStartHead(int, Event*)+0xd8 [traffic_server] (55e891616368)
  EThread::process_event(Event*, int)+0x276 [traffic_server] (55e89170b586)
  EThread::execute_regular()+0x33d [traffic_server] (55e89170bf0d)
  EThread::execute()+0x171 [traffic_server] (55e89170c361)
  spawn_thread_internal(void*)+0x55 [traffic_server] (55e89170a8a5)
  start_thread+0xc5 [libpthread-2.17.so] (7f1216cf8ea5)
```

```
int
CacheVC::openReadStartHead(int event, Event *e)
{
    ...
    {
        CACHE_TRY_LOCK(lock, vol->mutex, mutex->thread_holding);
        if (!lock.is_locked()) {
            VC_SCHED_LOCK_RETRY();
        }
    }
}
```

ailure count 6294

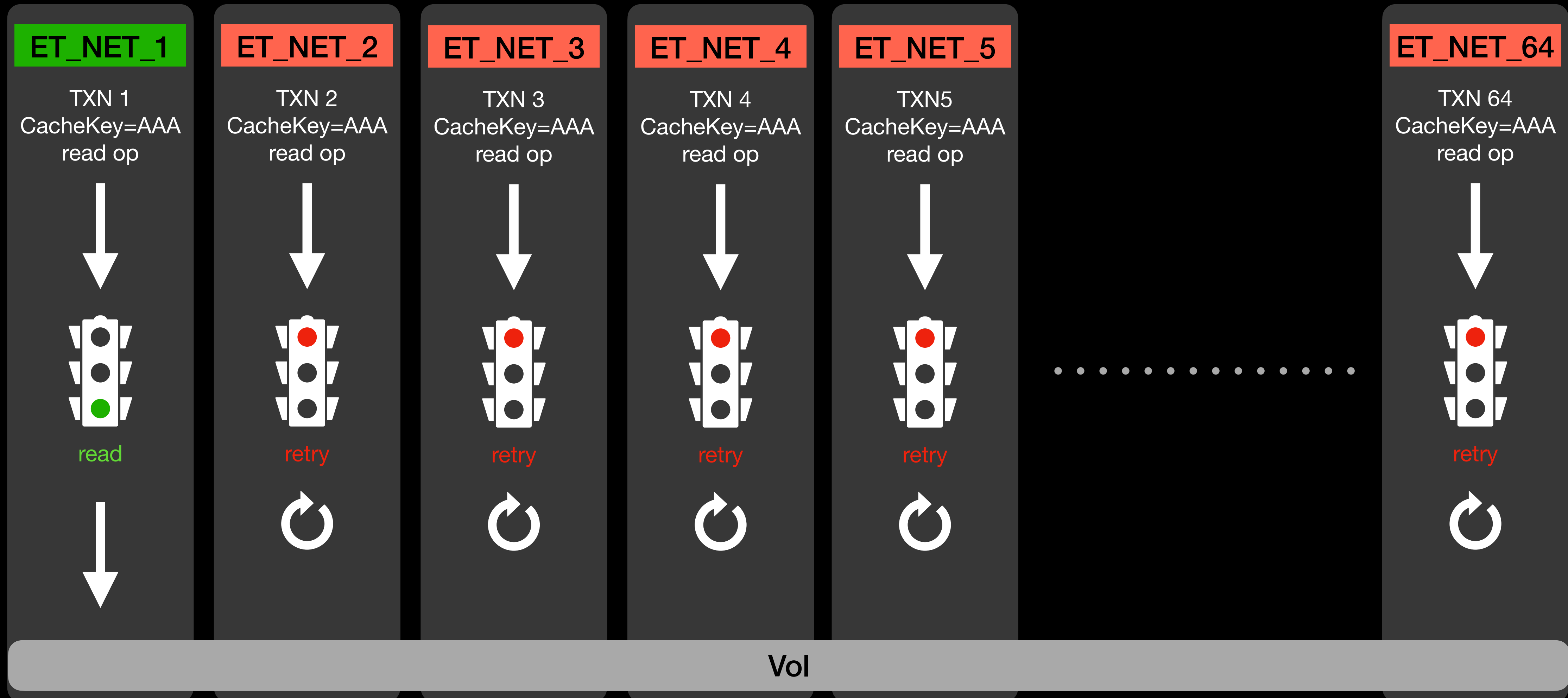
lure count 1391
rams const*, CacheFra
nfigParams const*, lo
onst*, long)+0x9d [tr

<https://github.com/apache/trafficserver/blob/c983006eccbce9365224c2cd30372528ac8df843/iocore/cache/CacheRead.cc#L1069>

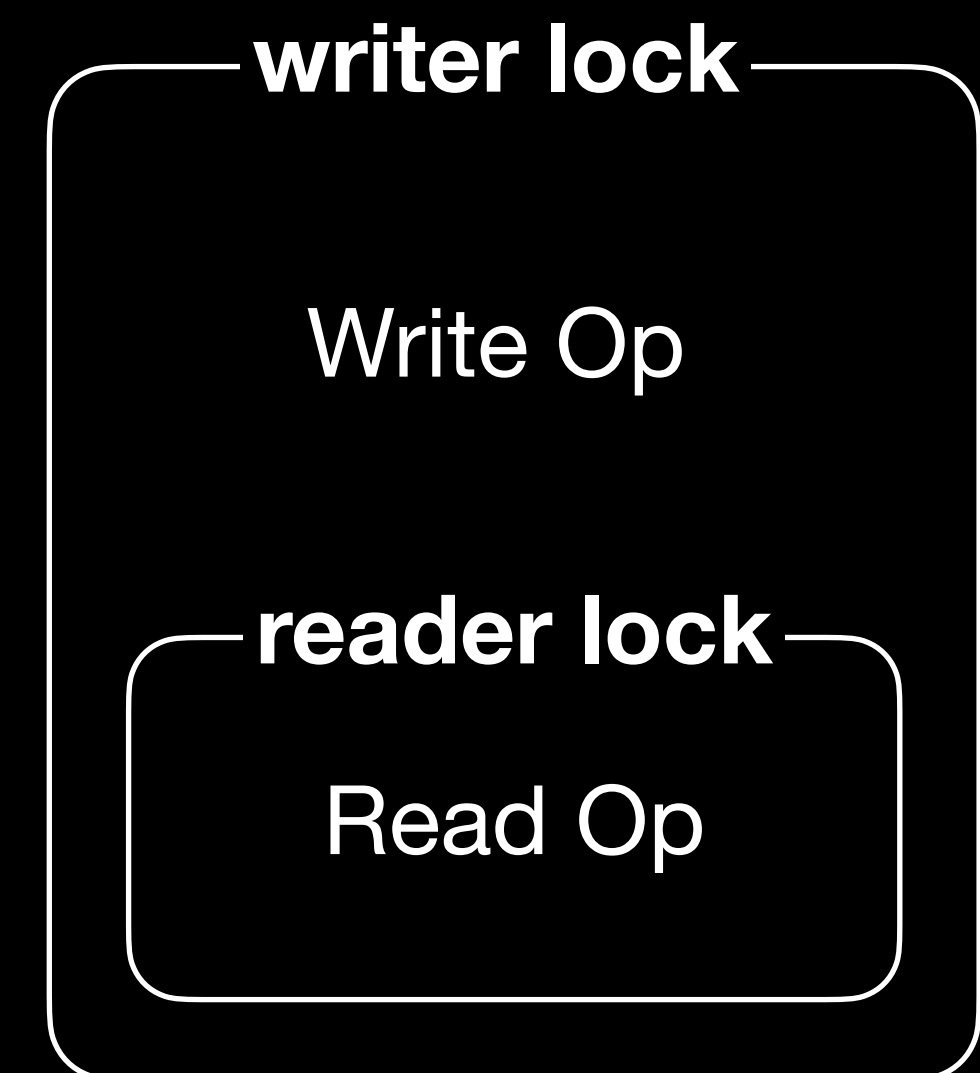
HttpSM::set_next_state()+0xb48 [traffic_server] (55e891482768)

HttpSM::set_next_state()+0xa06 [traffic_server] (55e891482626)

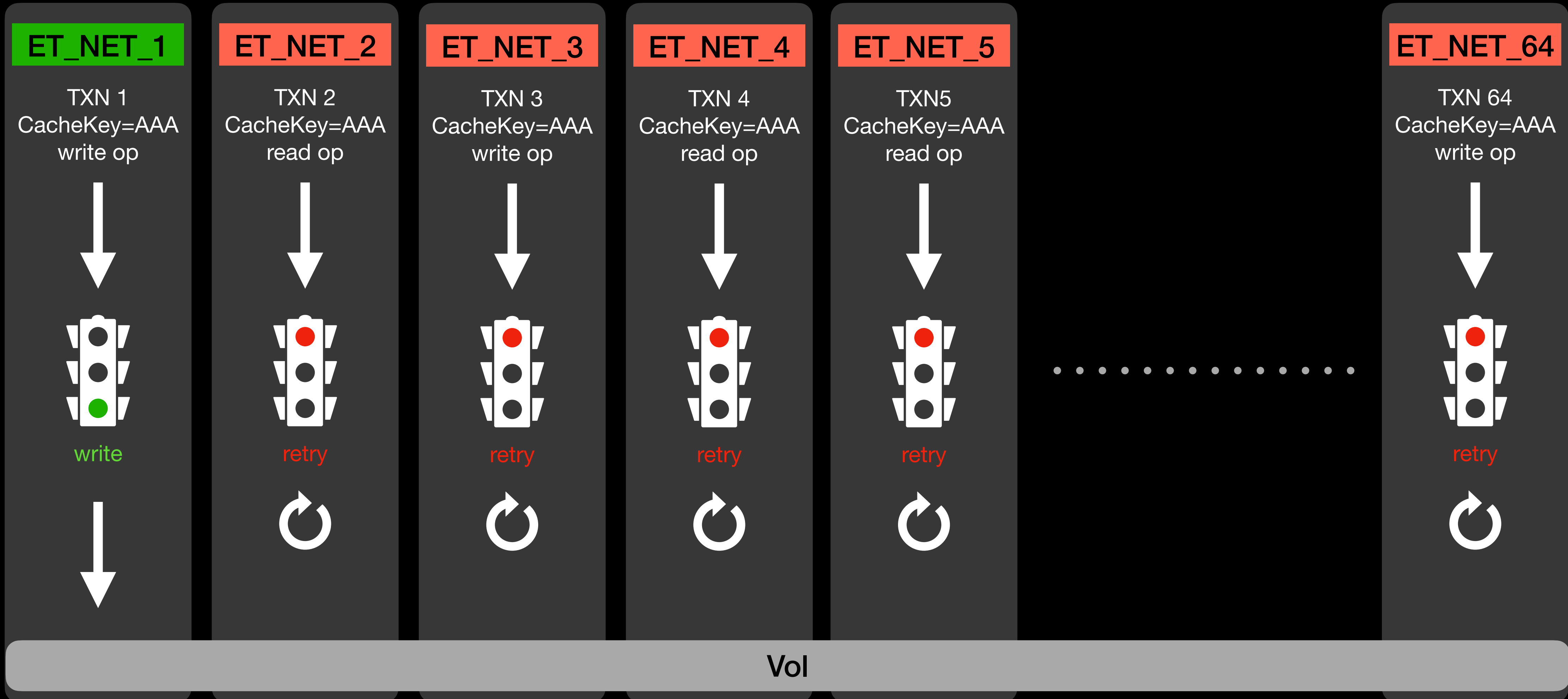
Try Lock Contention of Vol Mutex



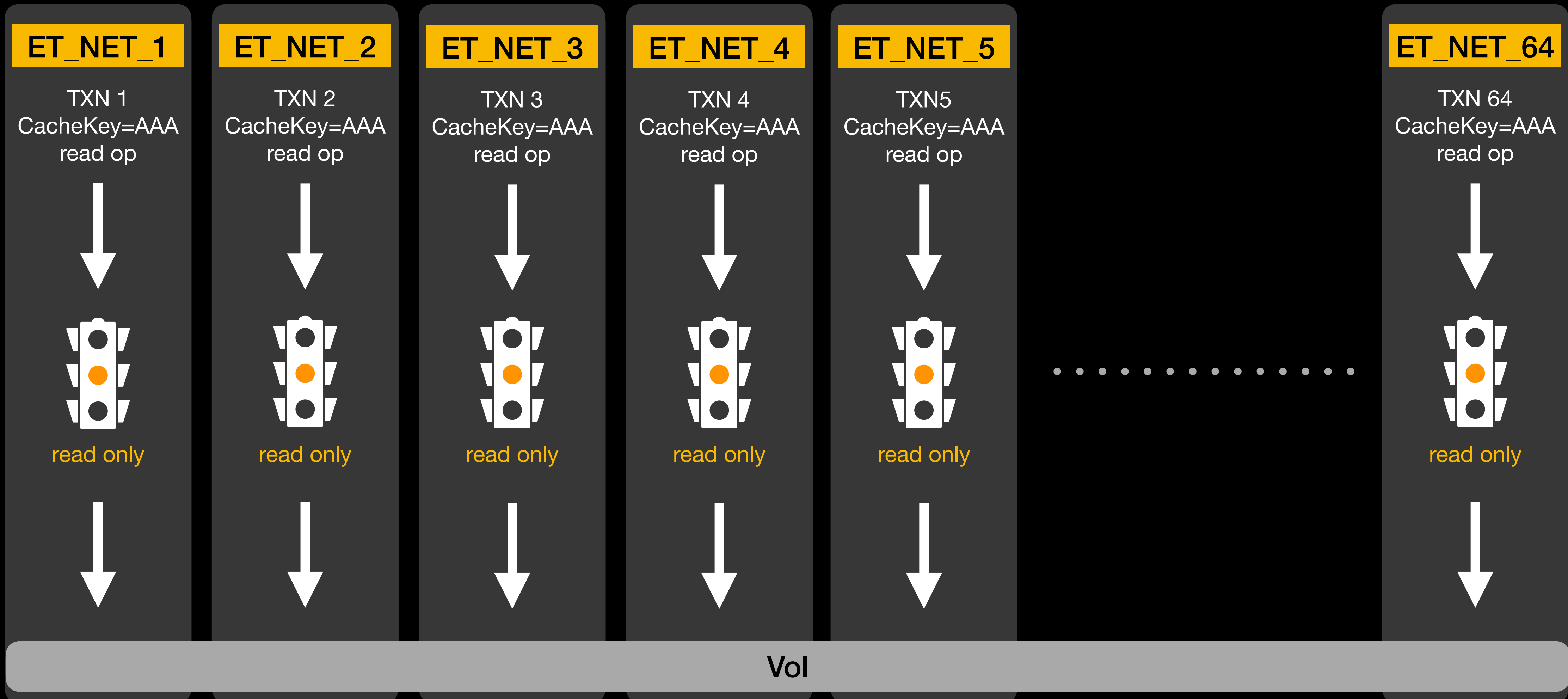
Reader-Writer Lock?



Reader-Writer Lock (Cache Write)

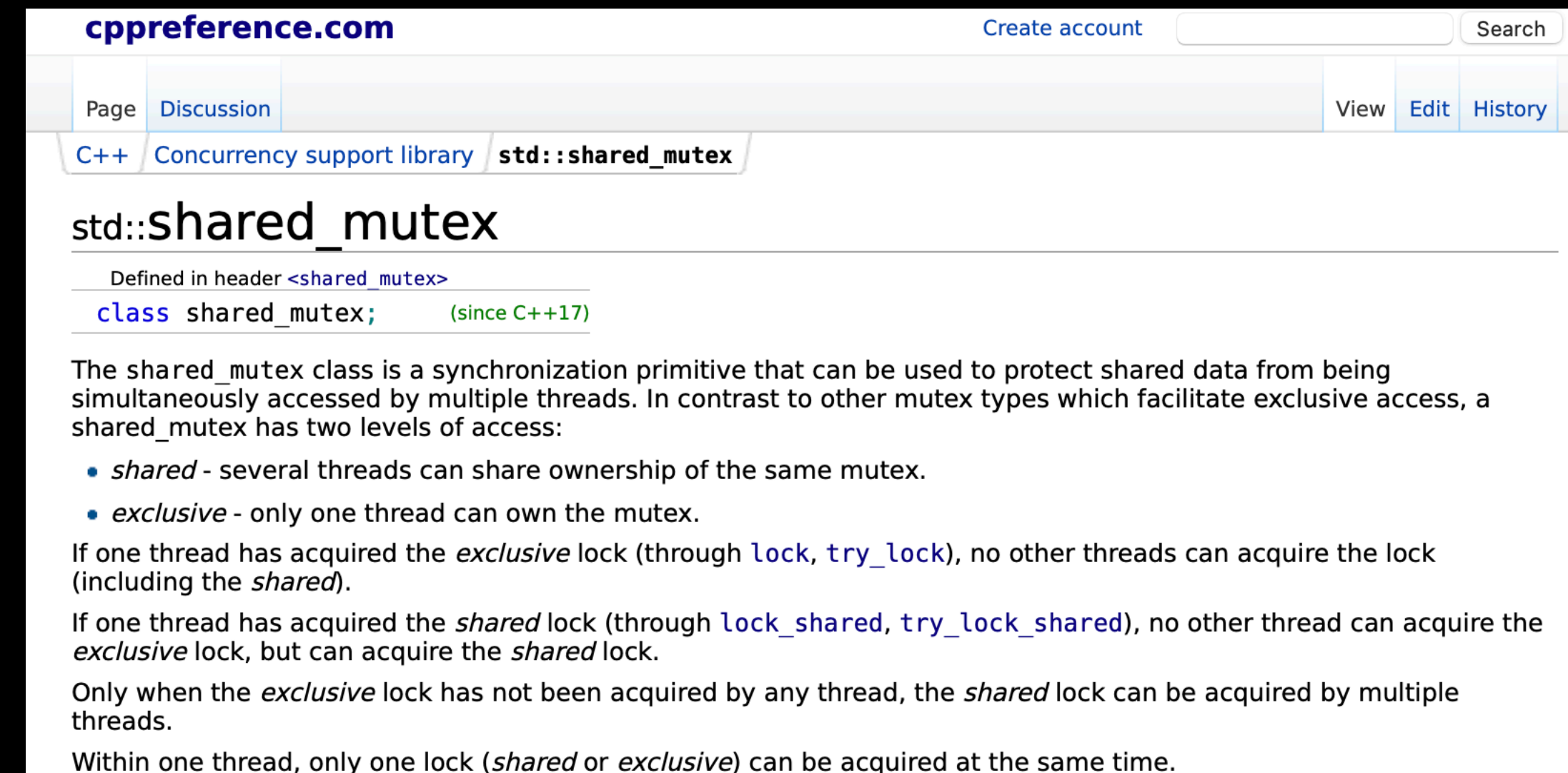


Reader-Writer Lock (Cache Read)



TrafficServer (9.2.0)

+ Replace Vol Mutex with `std::shared_mutex`



The screenshot shows the documentation for `std::shared_mutex` on the website `cppreference.com`. The page title is `std::shared_mutex`. It is defined in the header `<shared_mutex>`. The code snippet shows `class shared_mutex;` with a note that it is available since C++17. The text explains that `shared_mutex` is a synchronization primitive for shared data, offering two levels of access: `shared` (allowing multiple threads) and `exclusive` (allowing only one thread). It also notes that `exclusive` locks prevent `shared` locks and vice versa, and that only one lock can be held within a single thread.

cppreference.com [Create account](#)

Page [Discussion](#) [View](#) [Edit](#) [History](#)

[C++](#) [Concurrency support library](#) [std::shared_mutex](#)

std::shared_mutex

Defined in header `<shared_mutex>`

```
class shared_mutex; (since C++17)
```

The `shared_mutex` class is a synchronization primitive that can be used to protect shared data from being simultaneously accessed by multiple threads. In contrast to other mutex types which facilitate exclusive access, a `shared_mutex` has two levels of access:

- *shared* - several threads can share ownership of the same mutex.
- *exclusive* - only one thread can own the mutex.

If one thread has acquired the *exclusive* lock (through `lock`, `try_lock`), no other threads can acquire the lock (including the *shared*).

If one thread has acquired the *shared* lock (through `lock_shared`, `try_lock_shared`), no other thread can acquire the *exclusive* lock, but can acquire the *shared* lock.

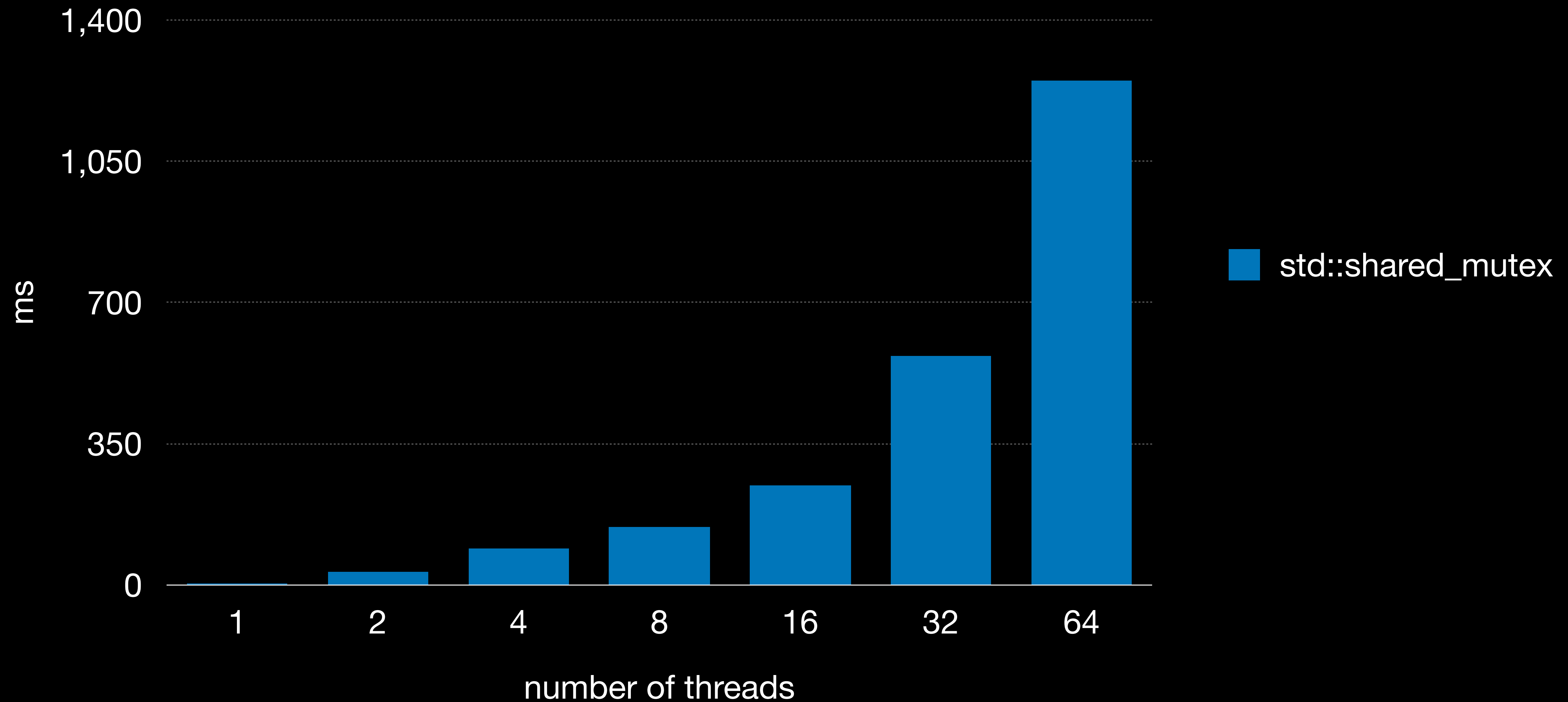
Only when the *exclusive* lock has not been acquired by any thread, the *shared* lock can be acquired by multiple threads.

Within one thread, only one lock (*shared* or *exclusive*) can be acquired at the same time.

Micro-Benchmark of `std::shared_mutex`

read-read performance

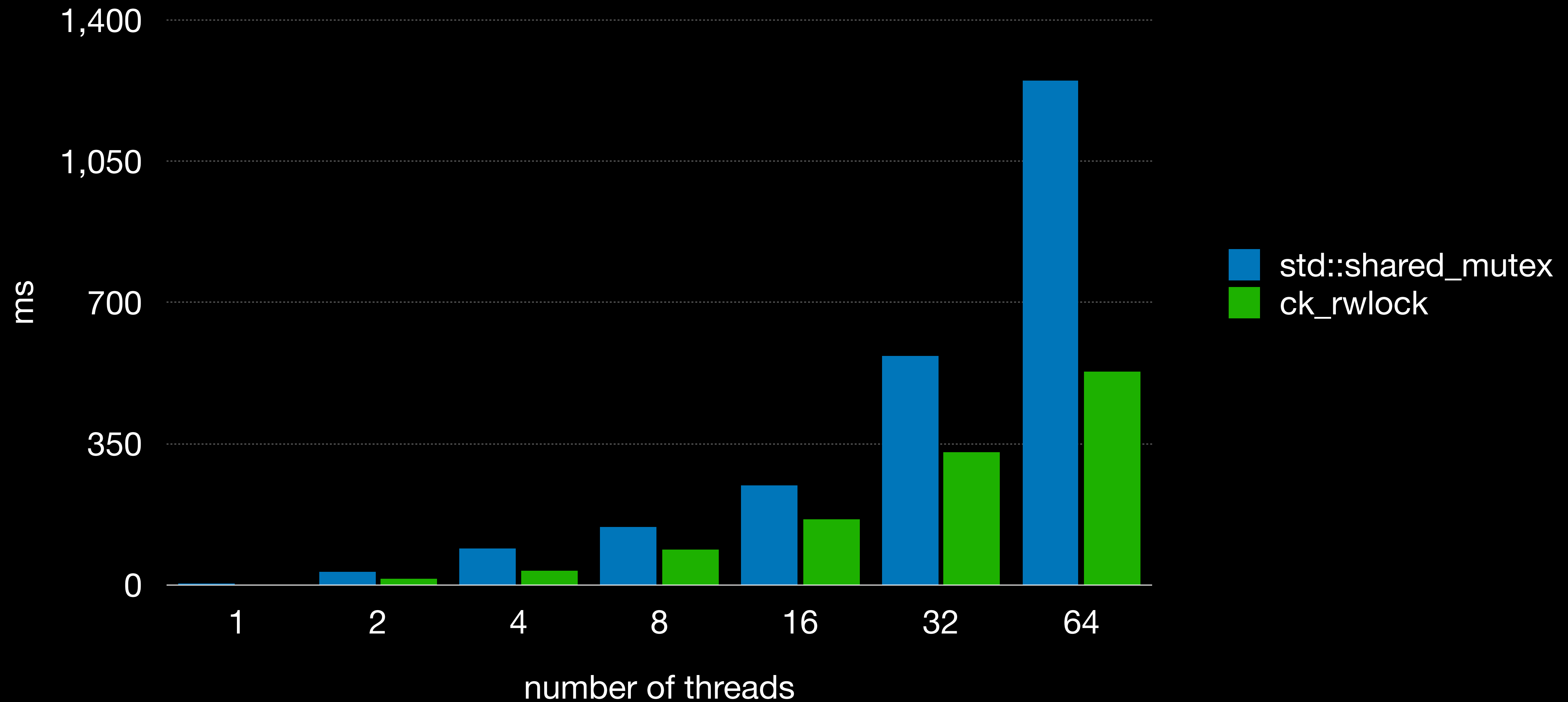
(completion time of 10,000 read op on each thread, lower is better)



Micro-Benchmark of ck_rwlock

read-read performance

(completion time of 10,000 read op on each thread, lower is better)




BRAVO - Biased Locking for Reader-Writer Locks

Dave Dice and Alen Kogan, Oracle Labs - USENIX'19

- > BRAVO acts as an accelerator layer, as readers can always
- > fall back to the traditional underlying lock to gain read access.
- > ...
- > Write performance and the scalability of read-vs-write and
- > write-vs-write behavior depends solely on the underlying lock.

- Section 3.



BRAVO—Biased Locking for Reader-Writer Locks
Dave Dice and Alex Kogan, *Oracle Labs*
<https://www.usenix.org/conference/atc19/presentation/dice>

This paper is included in the Proceedings of the
2019 USENIX Annual Technical Conference.
July 10–12, 2019 • Renton, WA, USA
ISBN 978-1-939133-03-8

Open access to the Proceedings of the
2019 USENIX Annual Technical Conference
is sponsored by USENIX.

BRAVO implementation in C++ (#9394)

```
template <typename T = std::shared_mutex, size_t SLOT_SIZE = 256>
class shared_mutex_impl
{
    struct alignas(hardware_constructive_interference_size) Slot
    {
        std::atomic<bool> mu = false;
    };

    struct Mutex {
        std::atomic<bool> read_bias = false;
        std::array<Slot, SLOT_SIZE> readers = {};
        time_point inhibit_until{};
        T underlying;
    };
};
```

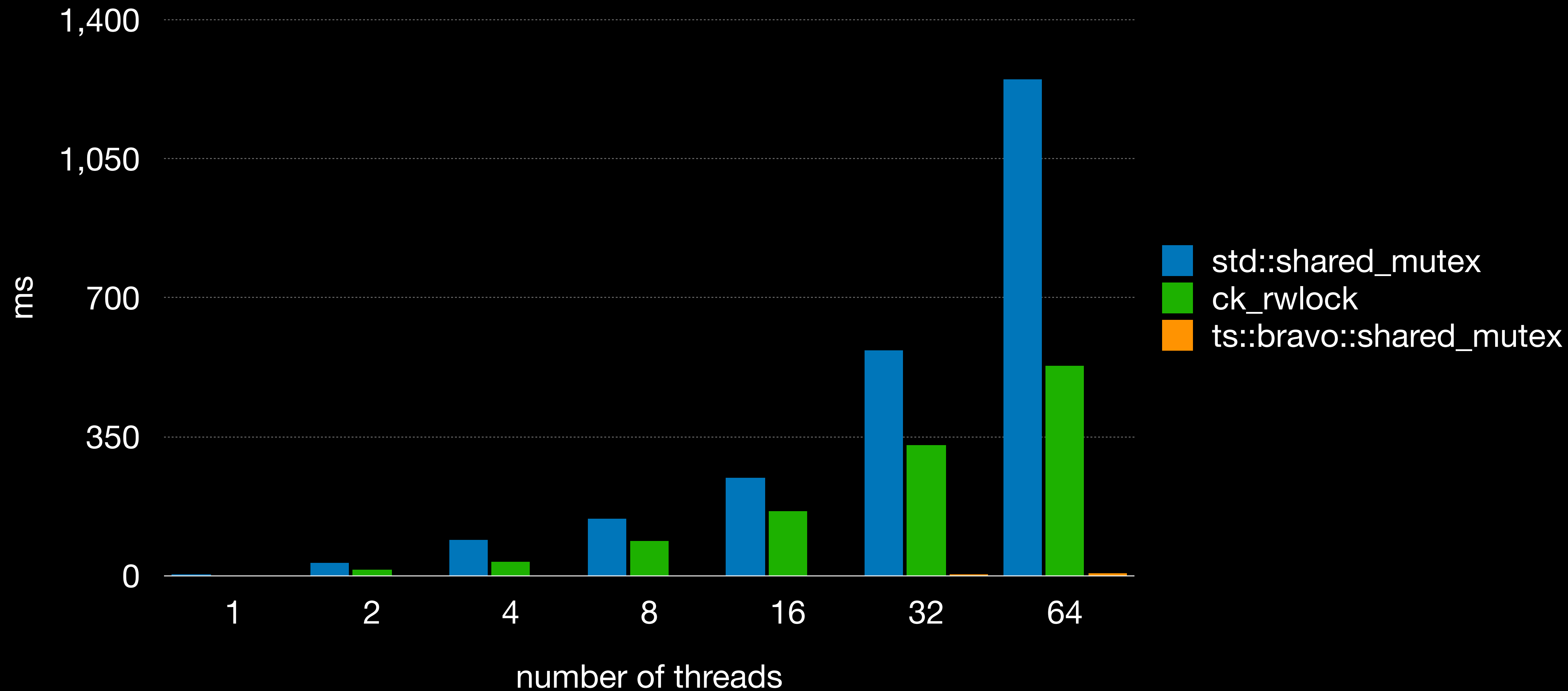
```
void
lock_shared(Token &token)
{
    // Fast path
    if (_mutex.read_bias.load(std::memory_order_acquire)) {
        size_t index = DenseThreadId::self() % SLOT_SIZE;
        Slot &slot = _mutex.readers[index];
        bool expect = false;
        if (slot.mu.compare_exchange_strong(expect, true, std::memory_order_acquire)) {
            // recheck
            if (_mutex.read_bias.load(std::memory_order_acquire)) {
                token = index + 1;
                return;
            } else {
                slot.mu.store(false, std::memory_order_relaxed);
            }
        }
    }

    // Slow path
    _mutex.underlying.lock_shared();
    if (_mutex.read_bias.load(std::memory_order_acquire) == false)
        _mutex.read_bias.store(true, std::memory_order_release);
}
}
```

Micro-Benchmark of BRAVO Implementation

read-read performance

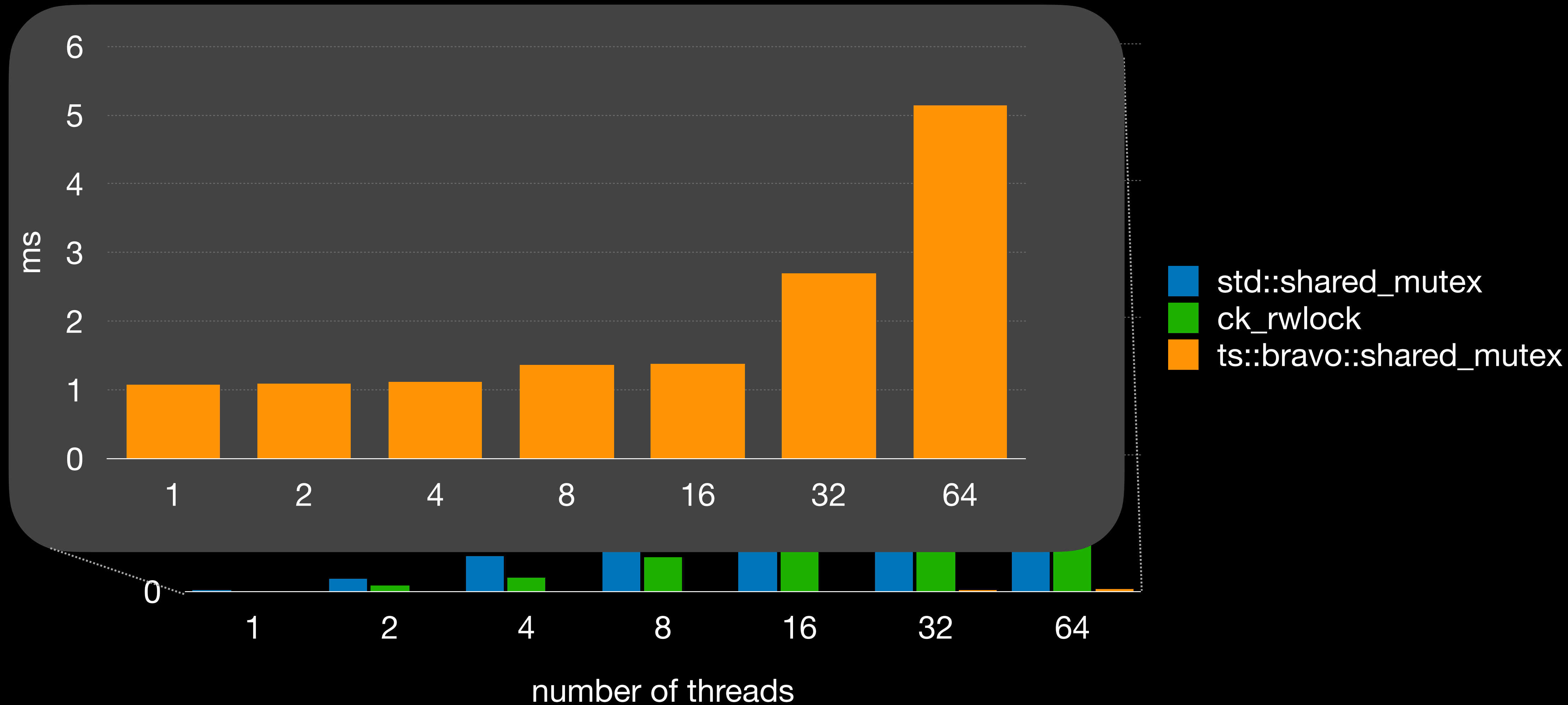
(completion time of 10,000 read op on each thread, lower is better)



Micro-Benchmark of BRAVO Implementation

read-read performance

(completion time of 10,000 read op on each thread, lower is better)



BRAVO Reader-Writer Lock Use Case and candidates

- librecords (ink_rwlock)
 - Use BRAVO lock for g_records_rwlock #9395
 - 5% improvement (554,388 -> 582,533 rps)
- HostDB (std::shared_mutex)
 - Introduced by Replace exclusive locks with rwlocks in hostdb #9442
- CacheHostTable (std::shared_mutex)
 - Introduced by Fix hosting.config reload #9046
 - std::shared_ptr might be better approach

TrafficServer (9.2.0)

- + Replace Vol Mutex with `std::shared_mutex`**
- + BRAVO reader-writer lock (#9394)**

Rule #1. No write op under reader lock

RAM Cache

Under holding reader lock

P_CacheVol.h

```
123 struct Vol : public Continuation {  
...  
155 RamCache *ram_cache = nullptr;
```

RAM Cache 1

RAM Cache 2



Vol 1

Vol 2

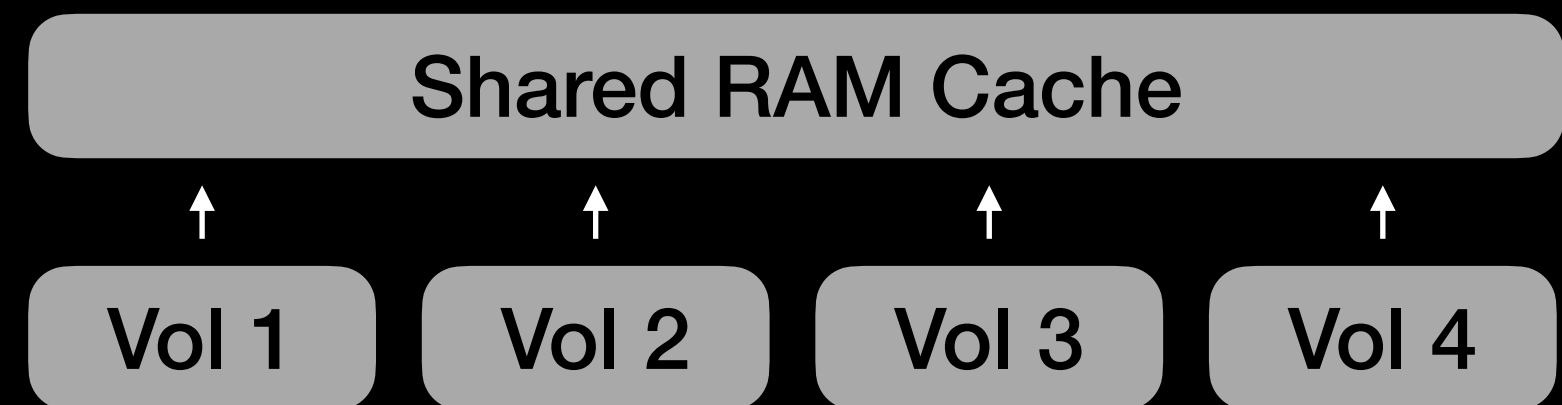
RamCacheLRU.cc

```
119 int  
120 RamCacheLRU::get(CryptoHash *key, Ptr<IOBufferData> *ret_data, uint64_t auxkey)  
121 {  
...  
125 uint32_t i = key->slice32(3) % nbuckets;  
126 RamCacheLRUEntry *e = bucket[i].head;  
127 while (e) {  
128     if (e->key == *key && e->auxkey == auxkey) {  
129         lru.remove(e);  
130         lru.enqueue(e);  
131         (*ret_data) = e->data;  
132         DDebug("ram_cache", "get %X %" PRIu64 " HIT", key->slice32(3), auxkey);  
133         CACHE_SUM_DYN_STAT_THREAD(cache_ram_cache_hits_stat, 1);  
134         return 1;  
135     }  
136     e = e->hash_link.next;  
137 }
```

Shared Lockless RAM Cache (#7351)

by John Plevyak <jplevyak@apache.org>

- Shared RAM Cache
- LRU with atomic operations



```
apache/trafficserver Public
Code Issues 332 Pull requests 33 Actions Projects 21 Wiki Security 12 Insights

Shared lockless RAM cache. #7351
Closed jplevyak wants to merge 5 commits into apache:master from jplevyak:sharedramcache

Conversation 3 Commits 5 Checks 0 Files changed 9 +529 -84
0 / 9 files viewed Review changes

Filter changed files
iocore/cache
Cache.cc
CacheDir.cc
CacheTest.cc
CacheWrite.cc
Makefile.am
P_RamCache.h
RamCacheCLFUS.cc
RamCacheLRU.cc
RamCacheLocklessLRU.cc

@@ -45,6 +45,8 @@ constexpr ts::VersionNumber CACHE_DB_VERSION(CACHE_DB_MAJOR_VERSION, CACHE_DB_MI
45 #define USELESS_REENABLES // allow them for now
46 // #define VERIFY_JTEST_DATA
47
48 + #define LOCKLESS_RAM_CACHE 1
49 +
48 static size_t DEFAULT_RAM_CACHE_MULTIPLIER = 10;
// I.e. 10x 1MB per 1GB of disk.
49
50 // This is the oldest version number that is
still usable.
51 // This is the oldest version number that is
still usable.

@@ -912,18 +914,22 @@ CacheProcessor::cacheInitialized()
912 Debug("cache_init",
"CacheProcessor::cacheInitialized -
caches_ready=0x%X, gnvols=%d", (unsigned
int)caches_ready,
913 gnvols.load());
914
915 - int64_t ram_cache_bytes = 0;
916 + int64_t ram_cache_bytes = 0;
917 + RamCache *lockless_ram_cache = nullptr;
918 + #ifdef LOCKLESS_RAM_CACHE
919 + lockless_ram_cache =
new_RamCacheLocklessLRU();
920 + #endif

916 if (gnvols) {
917 // new ram_caches, with algorithm from the
config
918 for (i = 0; i < gnvols; i++) {
919 switch (cache_config_ram_cache_algorithm)
{
920 default:
921 case RAM_CACHE_ALGORITHM_CLFUS:
922 - gvol[i]->ram_cache =
new_RamCacheCLFUS();
923 + gvol[i]->ram_cache = lockless_ram_cache
? lockless_ram_cache : new_RamCacheCLFUS();
924 break;
925 case RAM_CACHE_ALGORITHM_LRU:
926 - gvol[i]->ram_cache = new_RamCacheLRU();
927 + gvol[i]->ram_cache = lockless_ram_cache
? lockless_ram_cache : new_RamCacheLRU();
```

TrafficServer (9.2.0)

- + Replace Vol Mutex with `std::shared_mutex`**
- + BRAVO reader-writer lock (#9394)**
- + Shared Lockless RAM Cache (#7351)**

64 threads

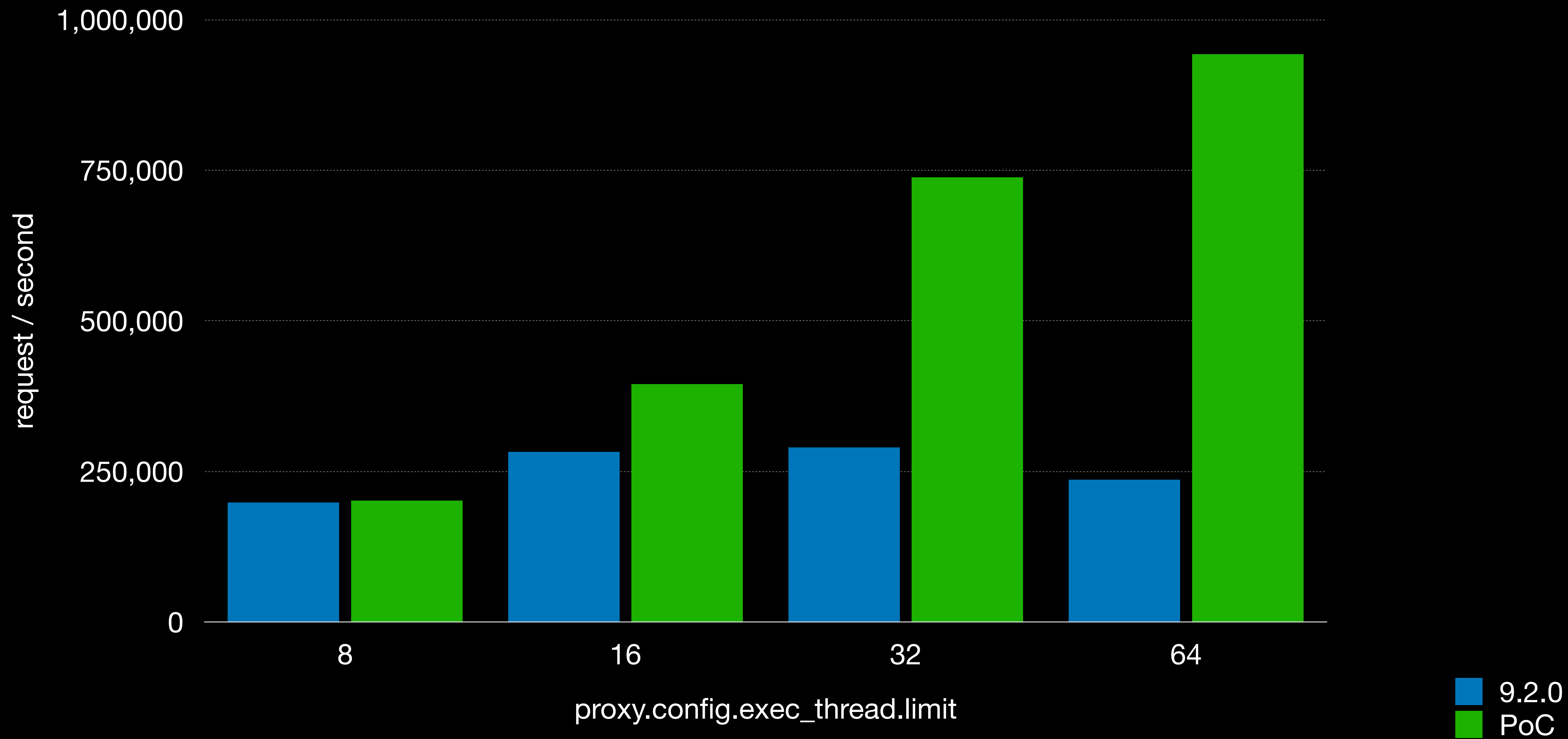
9.2.0

```
1 [||||| 10.3%] 17 [|| 3.3%] 33 [||||| 14.5%] 49 [||||||| 25.3%]
2 [||||| 9.3%] 18 [||| 8.1%] 34 [||||||| 21.6%] 50 [|| 3.2%]
3 [||||||| 19.1%] 19 [||| 5.9%] 35 [||||||| 28.9%] 51 [||||||| 16.4%]
4 [||| 6.5%] 20 [||||||| 38.9%] 36 [||| 7.3%] 52 [|| 3.4%]
5 [||||||| 25.8%] 21 [||||||| 46.1%] 37 [||||||| 44.7%] 53 [||| 9.9%]
6 [| 2.0%] 22 [||||||| 19.9%] 38 [||||||| 16.9%] 54 [||||||| 74.2%]
7 [||| 5.2%] 23 [||||||| 47.7%] 39 [||||||| 16.4%] 55 [|| 3.9%]
8 [||||||| 22.1%] 24 [||||||| 44.0%] 40 [||||||| 32.2%] 56 [||||||| 40.8%]
9 [||||||| 19.3%] 25 [||||||| 16.2%] 41 [||||||| 35.1%] 57 [||||||| 20.3%]
10 [||||||| 64.1%] 26 [| 1.3%] 42 [||||| 10.9%] 58 [||||||| 30.7%]
11 [||| 7.2%] 27 [||||| 9.5%] 43 [||||||| 49.3%] 59 [||||||| 14.8%]
12 [||||||| 63.8%] 28 [||||||| 30.9%] 44 [||||| 11.6%] 60 [| 3.3%]
13 [||||| 10.2%] 29 [| 1.3%] 45 [||||||| 21.3%] 61 [||||||| 24.2%]
14 [|| 4.6%] 30 [||||||| 27.5%] 46 [||||||| 35.3%] 62 [||||||| 22.9%]
15 [||||||| 20.8%] 31 [||||||| 49.0%] 47 [| 4.6%] 63 [||| 6.5%]
16 [||||||| 28.8%] 32 [||||||| 18.9%] 48 [||||| 13.2%] 64 [| 3.9%]
Mem[||||||| 7.23G/376G] Tasks: 119, 577 thr; 13 running
Swp[ 0K/23.4G] Load average: 4.88 2.36 1.08
Uptime: 62 days, 21:14:52
```

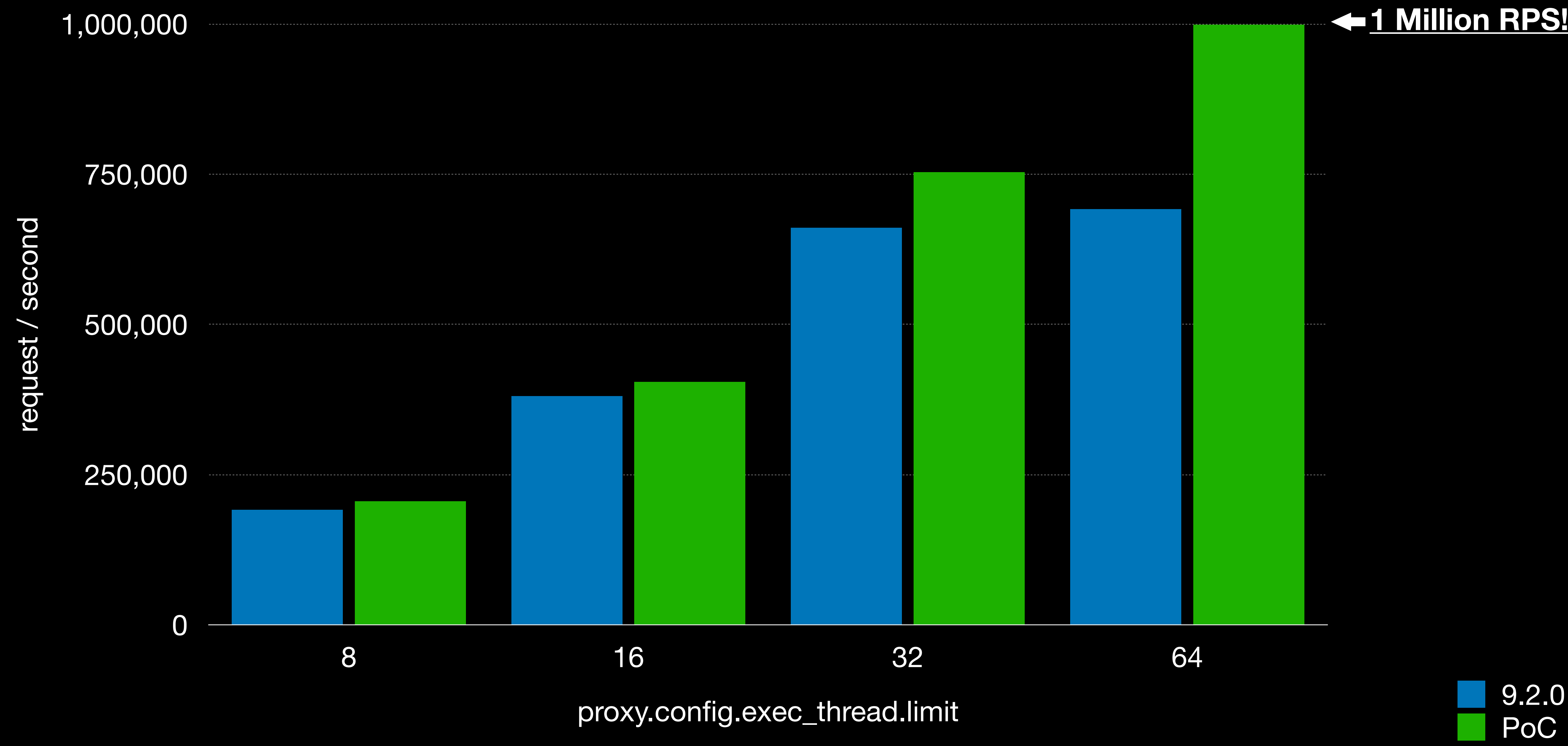
PoC

```
1 [||||||| 96.1%] 17 [||||||| 100.0%] 33 [||||||| 93.4%] 49 [||||||| 98.1%]
2 [||||||| 85.4%] 18 [||||||| 92.8%] 34 [||||||| 96.1%] 50 [||||||| 81.9%]
3 [||||||| 90.2%] 19 [||||||| 97.4%] 35 [||||||| 96.8%] 51 [||||||| 91.5%]
4 [||||||| 87.5%] 20 [||||||| 98.1%] 36 [||||||| 92.2%] 52 [||||||| 97.4%]
5 [||||||| 96.1%] 21 [||||||| 94.2%] 37 [||||||| 96.7%] 53 [||||||| 100.0%]
6 [||||||| 98.7%] 22 [||||||| 100.0%] 38 [||||||| 99.4%] 54 [||||||| 98.0%]
7 [||||||| 87.0%] 23 [||||||| 91.4%] 39 [||||||| 94.8%] 55 [||||||| 82.1%]
8 [||||||| 96.1%] 24 [||||||| 100.0%] 40 [||||||| 94.8%] 56 [||||||| 100.0%]
9 [||||||| 97.4%] 25 [||||||| 100.0%] 41 [||||||| 96.8%] 57 [||||||| 93.5%]
10 [||||||| 98.1%] 26 [||||||| 99.4%] 42 [||||||| 98.1%] 58 [||||||| 100.0%]
11 [||||||| 98.7%] 27 [||||||| 89.6%] 43 [||||||| 97.4%] 59 [||||||| 83.8%]
12 [||||||| 87.3%] 28 [||||||| 83.3%] 44 [||||||| 92.9%] 60 [||||||| 98.7%]
13 [||||||| 96.7%] 29 [||||||| 99.4%] 45 [||||||| 92.7%] 61 [||||||| 100.0%]
14 [||||||| 98.7%] 30 [||||||| 97.4%] 46 [||||||| 96.1%] 62 [||||||| 96.8%]
15 [||||||| 96.1%] 31 [||||||| 100.0%] 47 [||||||| 96.7%] 63 [||||||| 98.1%]
16 [||||||| 98.7%] 32 [||||||| 100.0%] 48 [||||||| 98.7%] 64 [||||||| 99.4%]
Mem[||||||| 7.40G/376G] Tasks: 119, 577 thr; 64 running
Swp[ 0K/23.4G] Load average: 20.64 13.70 6.42
Uptime: 62 days, 21:35:33
```

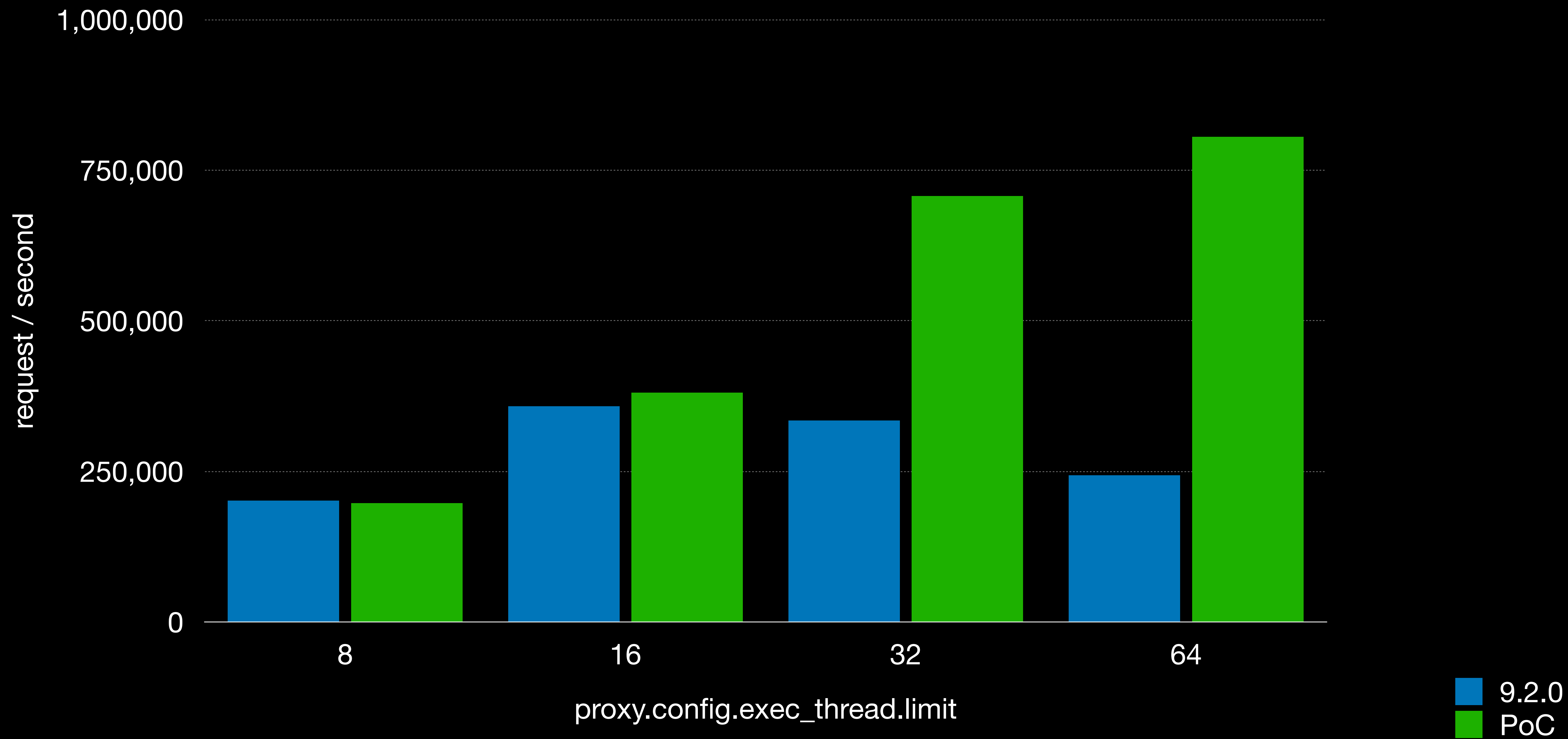
URL=1, Content Size=0B



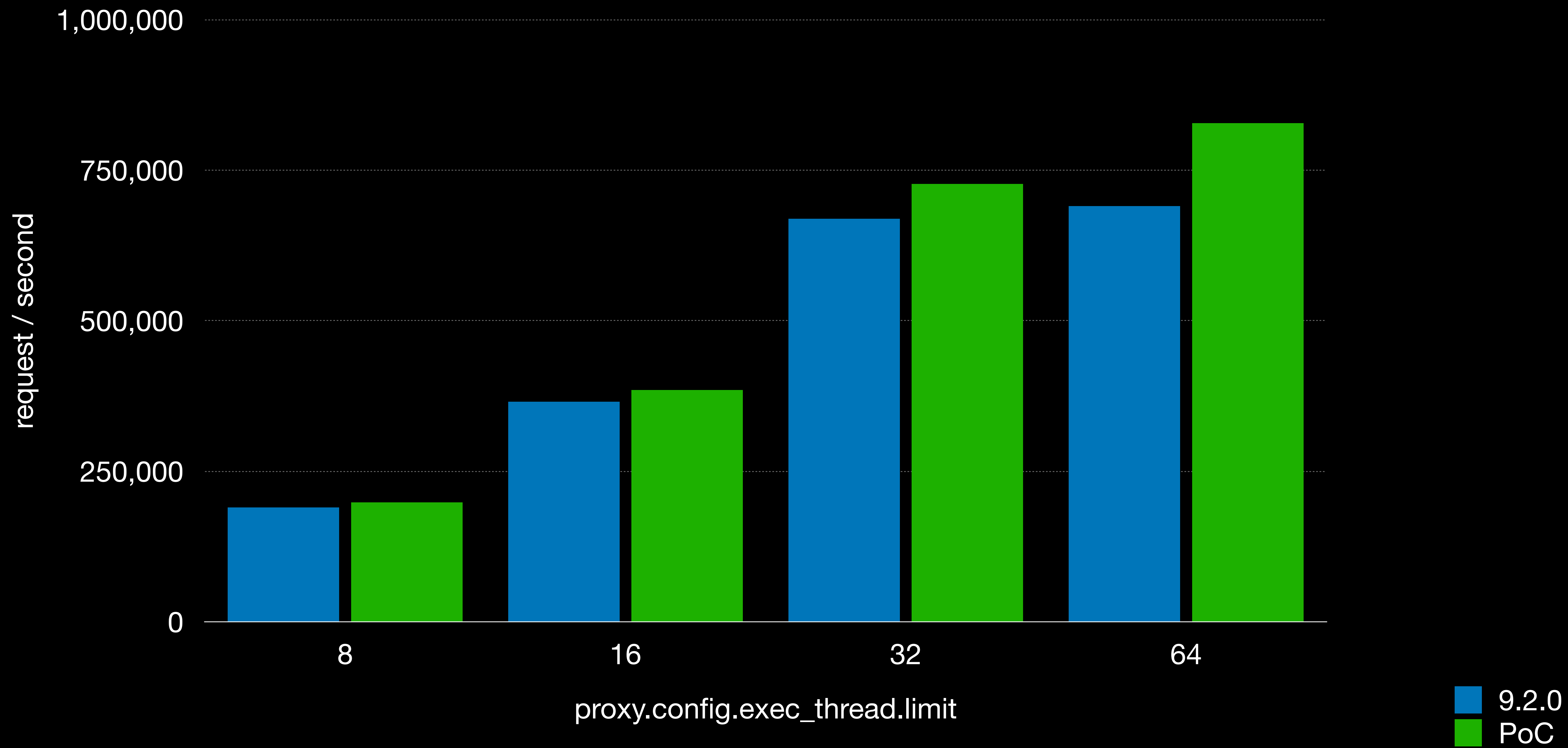
URL=64, Content Size=0B, volume=10



URL=1, Content Size=1KB



URL=64, Content Size=1KB, volume=10



Can you deploy on production!?

Challenges

Adjust current code with reader lock

CacheRead.cc

```
524 int
525 CacheVC::openReadClose(int event, Event * /* e_ATS_UNUSED */)
525 {
...
534     CACHE_TRY_LOCK(lock, vol->mutex, mutex->thread_holding);
535     if (!lock.is_locked()) {
536         VC_SCHED_LOCK_RETRY();
537     }
538     if (f.hit_evacuate && dir_valid(vol, &first_dir) && closed > 0) {
539         if (f.single_fragment) {
540             vol->force_evacuate_head(&first_dir, dir_pinned(&first_dir));
541         } else if (dir_valid(vol, &earliest_dir)) {
542             vol->force_evacuate_head(&first_dir, dir_pinned(&first_dir));
543             vol->force_evacuate_head(&earliest_dir, dir_pinned(&earliest_dir));
544         }
545     }
```

Challenges

Adjust current code with reader lock

CacheDir.cc

```
535 int
536 dir_probe(const CacheKey *key, Vol *vol, Dir *result, Dir **last_collision)
537 {
...
549     e = dir_bucket(b, seg);
...
570     if (dir_valid(vol, e)) {
...
577     } else { // delete the invalid entry
578         CACHE_DEC_DIR_USED(vol->mutex);
579         e = dir_delete_entry(e, p, s, vol);
580         continue;
581     }
```

Summary

- **PoC: TrafficServer (9.2.0)**
 - + **Replace Vol Mutex with `std::shared_mutex`**
 - + **BRAVO reader-writer lock (#9394)**
 - + **Shared Lockless RAM Cache (#7351)**
- **More works to adjust code with reader lock is required**
- **BRAVO reader-writer lock can apply read heavy cases**

Apache Traffic Server

Related Issue

cache_dir_sync triggers an increase in cache read time on ATS 9.1.3 #9124

- Root cause is Try Lock Contention of Vol Mutex
- Increasing number of volumes is mitigation (1 -> 5)

```
CacheDir.cc
```

```
1072 int
1073 CacheSync::mainEvent(int event, Event *e)
1074 {
    ...
1117     CACHE_TRY_LOCK(lock, gvol[vol_idx]->mutex, mutex->thread_holding);
1118     if (!lock.is_locked()) {
1119         trigger = eventProcessor.schedule_in(this, HRTIME_MSECONDS(cache_config_mutex_retry_
1120         return EVENT_CONT;
1121     }
```


Benchmark Conditions

Client

- Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz 64 Core
- wrk2, 64 threads

```
» taskset -c 0-63 wrk2 -t 64 -c 640 -d 10 -R 2000000 http://targetbox/static/0B
```

```
» taskset -c 0-63 wrk2 -t 64 -c 640 -d 10 -R 2000000 http://targetbox/static/1KB
```

Benchmark Conditions

ATS 9.2.0

- Intel(R) Xeon(R) Gold 5218 CPU @ 2.30GHz 64 Core
- jemalloc 5.3.0

```
» traffic_server -fF
```

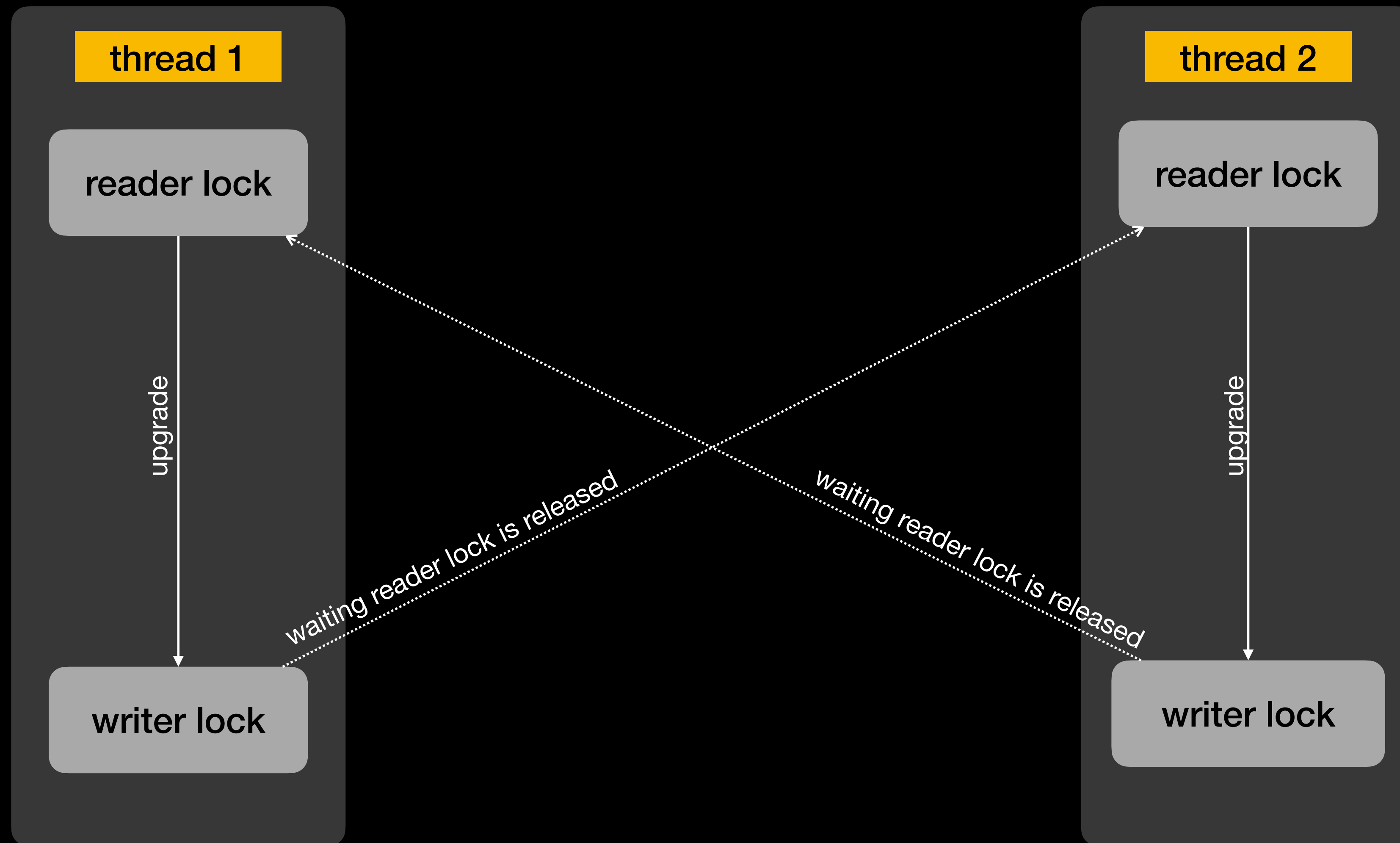
- records.config

```
CONFIG proxy.config.accept_threads INT 0
CONFIG proxy.config.exec_thread.listen INT 1
CONFIG proxy.config.exec_thread.autoconfig INT 0
CONFIG proxy.config.exec_thread.limit INT 64 # 0,1,2,4,8,16,32,64
CONFIG proxy.config.exec_thread.affinity INT 4

CONFIG proxy.config.cache.ram_cache.size INT 104857600 # 100MB

CONFIG proxy.config.http.insert_response_via_str INT 2
```

Dead-lock of upgrading



32 threads

9.2.0

```
1 [|||||] 40.8% 17 [||] 3.4% 33 [|||||] 56.6% 49 [ ] 0.0%
2 [|||||] 30.8% 18 [||] 6.3% 34 [|||||] 74.3% 50 [||] 2.0%
3 [|||||] 57.2% 19 [||||] 11.4% 35 [|||||] 76.8% 51 [||] 1.4%
4 [||||] 12.5% 20 [||||] 8.4% 36 [|||||] 78.9% 52 [||] 2.8%
5 [|||||] 84.9% 21 [||] 1.3% 37 [|||||] 49.3% 53 [||] 2.0%
6 [|||||] 28.9% 22 [ ] 0.0% 38 [|||||] 43.0% 54 [||] 4.6%
7 [|||||] 23.7% 23 [ ] 0.0% 39 [|||||] 59.9% 55 [ ] 0.0%
8 [|||||] 31.5% 24 [||] 0.7% 40 [|||||] 34.2% 56 [||] 0.7%
9 [|||||] 92.2% 25 [||] 6.2% 41 [|||||] 36.1% 57 [||] 4.2%
10 [|||||] 42.4% 26 [||] 6.4% 42 [|||||] 61.2% 58 [||] 6.2%
11 [|||||] 54.9% 27 [||] 2.9% 43 [|||||] 30.2% 59 [||] 2.1%
12 [|||||] 80.9% 28 [||] 1.4% 44 [||||] 12.4% 60 [||] 1.3%
13 [|||||] 49.3% 29 [||] 2.7% 45 [|||||] 60.9% 61 [||] 2.0%
14 [|||||] 46.8% 30 [||] 6.2% 46 [|||||] 55.6% 62 [||] 1.3%
15 [|||||] 40.3% 31 [||] 0.7% 47 [|||||] 23.2% 63 [ ] 0.0%
16 [|||||] 74.3% 32 [||||] 8.2% 48 [|||||] 47.7% 64 [||] 1.4%
Mem[|||||] 7.18G/376G Tasks: 116, 544 thr; 18 running
Swp[ ] 0K/23.4G Load average: 5.71 3.22 1.65
Uptime: 62 days, 21:18:56
```

PoC

```
1 [|||||] 100.0% 17 [||||] 12.2% 33 [|||||] 100.0% 49 [||||] 7.4%
2 [|||||] 100.0% 18 [||||] 10.3% 34 [|||||] 100.0% 50 [ ] 0.0%
3 [|||||] 100.0% 19 [||] 2.2% 35 [|||||] 100.0% 51 [||] 0.7%
4 [|||||] 100.0% 20 [||] 3.1% 36 [|||||] 100.0% 52 [||] 1.3%
5 [|||||] 100.0% 21 [||] 1.4% 37 [|||||] 100.0% 53 [||] 0.7%
6 [|||||] 100.0% 22 [||] 1.4% 38 [|||||] 100.0% 54 [||] 3.7%
7 [|||||] 100.0% 23 [||] 0.7% 39 [|||||] 100.0% 55 [||] 0.7%
8 [|||||] 100.0% 24 [||] 3.1% 40 [|||||] 100.0% 56 [||] 3.3%
9 [|||||] 100.0% 25 [||] 2.9% 41 [|||||] 100.0% 57 [||||] 8.5%
10 [|||||] 100.0% 26 [||||] 7.3% 42 [|||||] 100.0% 58 [||] 2.9%
11 [|||||] 100.0% 27 [||||] 7.6% 43 [|||||] 100.0% 59 [||||] 10.2%
12 [|||||] 100.0% 28 [||||] 8.7% 44 [|||||] 100.0% 60 [||] 3.8%
13 [|||||] 98.1% 29 [||] 4.7% 45 [|||||] 100.0% 61 [||] 0.7%
14 [|||||] 100.0% 30 [||] 3.8% 46 [|||||] 100.0% 62 [ ] 0.0%
15 [|||||] 100.0% 31 [ ] 0.0% 47 [|||||] 100.0% 63 [||||] 7.9%
16 [|||||] 100.0% 32 [||||] 9.9% 48 [|||||] 100.0% 64 [||||] 8.6%
Mem[|||||] 7.27G/376G Tasks: 119, 544 thr; 33 running
Swp[ ] 0K/23.4G Load average: 17.08 13.70 6.76
Uptime: 62 days, 21:36:14
```