Accelerating Traffic Server
ATS Spring 2023 Summit

Masaori Koshiba (masaori@apache.org)
Q: Does ATS scale with the number of cores?

**Conditions**
- Intel Xeon Gold 64 Core
- jemalloc-5.3.0 (no freelist)
- SO_REUSE_PORT (no accept thread)
- CPU Affinity 4 (assign threads to processing units)
- Target URL is only 1
- Header only response
- 100% RAM Cache Hit

![Graph showing request/second vs. proxy.config.exec_thread.limit with three curves labeled A, B, and C. The graph indicates that curve A represents 9.2.0.](image-url)
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?
### 16 threads

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |}
"Underutilized" Issue

64 threads

Tasks: 119, 577 thr; 13 running
Load average: 4.88 2.36 1.08
Uptime: 62 days, 21:14:52
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*, CacheFragType, char const*, int)+0x10a [traffic_server] (55e89161845a)
  CacheProcessor::open_read(Continuation*, HttpCacheKey const*, HTTPHdr*, OverridableHttpConfigParams const*, long, CacheFragType)+0x27 [traffic_server] (55e8915fccf7)
  HttpCacheSM::open_read(HttpCacheKey const*, URL*, HTTPHdr*, OverridableHttpConfigParams const*, long)+0x9d [traffic_server] (55e89148b4cd)
  HttpSM::do_cache_lookup_and_read()+0x169 [traffic_server] (55e89147d929)
  HttpSM::set_next_state()+0xb48 [traffic_server] (55e891482768)
  HttpSM::set_next_state()+0xa06 [traffic_server] (55e891482626)
  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)
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*, CacheFragType, char const*, int)+0x10a [traffic_server] (55e89161845a)
CacheProcessor::open_read(Continuation*, HttpCacheKey const*, HTTPHdr*, OverridableHttpConfigParams const*, long, CacheFragType)+0x27 [traffic_server] (55e8915fccf7)
HttpCacheSM::open_read(HttpCacheKey const*, URL*, HTTPHdr*, OverridableHttpConfigParams const*, long)+0x9d [traffic_server] (55e89148b4cd)
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)

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

Fall 2022 Summit
Try Lock Contention of Vol Mutex

ET_NET_1
TXN 1
CacheKey=AAA
read op
read
retry
Vol

ET_NET_2
TXN 2
CacheKey=AAA
read op
retry
retry
Vol

ET_NET_3
TXN 3
CacheKey=AAA
read op
retry
retry
Vol

ET_NET_4
TXN 4
CacheKey=AAA
read op
retry
retry
Vol

ET_NET_5
TXN 5
CacheKey=AAA
read op
retry
retry
Vol

ET_NET_64
TXN 64
CacheKey=AAA
read op
retry
retry
Vol
Reader-Writer Lock?
Reader-Writer Lock (Cache Write)

- **ET_NET_1**: TXN 1, CacheKey=AAA, write op
- **ET_NET_2**: TXN 2, CacheKey=AAA, read op
- **ET_NET_3**: TXN 3, CacheKey=AAA, write op
- **ET_NET_4**: TXN 4, CacheKey=AAA, read op
- **ET_NET_5**: TXN 5, CacheKey=AAA, read op
- **ET_NET_64**: TXN 64, CacheKey=AAA, write op

Operations:
- write
- retry
- read op

Note: The diagram shows multiple transactions and their operations, with some transactions overlapping in their attempts to access the cache.
Reader-Writer Lock (Cache Read)

- **ET_NET_1**
  - TXN 1
  - CacheKey=AAA
  - read op
  - read only

- **ET_NET_2**
  - TXN 2
  - CacheKey=AAA
  - read op
  - read only

- **ET_NET_3**
  - TXN 3
  - CacheKey=AAA
  - read op
  - read only

- **ET_NET_4**
  - TXN 4
  - CacheKey=AAA
  - read op
  - read only

- **ET_NET_5**
  - TXN 5
  - CacheKey=AAA
  - read op
  - read only

- **ET_NET_64**
  - TXN 64
  - CacheKey=AAA
  - read op
  - read only

Vol
TrafficServer (9.2.0)
+ Replace Vol Mutex with std::shared_mutex
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 ops on each thread, lower is better)

std::shared_mutex
ck_rwlock

number of threads

ms

std::shared_mutex
ck_rwlock

1
2
4
8
16
32
64

0
350
700
1,050
1,400
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 implementation in C++ (#9394)

```cpp
template <typename T = std::shared_mutex, size_t SLOT_SIZE = 2>
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;
    };

    Mutex _mutex;

    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_relaxed)) {
                // recheck
                if (_mutex.read_bias.load(std::memory_order_acquire)) {
                    token = index + 1;
                    return;
                } else {
                    slot.mu.store(false, std::memory_order_relaxed);
                }
            } 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 {
124  ...
155  RamCache *ram_cache = nullptr;

RamCacheLRU.cc
119  int
120  RamCacheLRU::get(CryptoHash *key, Ptr<IOBufferData> *ret_data, uint64_t auxkey)
121  {
122  ...
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  }

Vol 1  RAM Cache 1  Vol 2  RAM Cache 2
Shared Lockless RAM Cache (#7351)
by John Plevyak <jplevyak@apache.org>

- Shared RAM Cache
- LRU with atomic operations
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

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

1 Million RPS!
URL=1, Content Size=1KB

![Bar graph showing the relationship between proxy.config.exec_thread.limit and request/second. The x-axis represents the number of proxy.config.exec_thread.limit values with bars for 8, 16, 32, and 64. The y-axis represents the request/second ranging from 0 to 1,000,000. The blue bars represent 9.2.0, and the green bars represent PoC. The peak is at 64 with 750,000 requests/second.](image-url)
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 }
Challenges
Adjust current code with reader lock

CacheDir.cc

535 int dir_probe(const CacheKey *key, Vol *vol, Dir *result, Dir **last_collision) {
... 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_delay);
1120 return EVENT_CONT;
1121 }
Benchmark Conditions

Client

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

```sh
» 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

Thread 1:
- Reader lock
- Upgrade
- Writer lock

Thread 2:
- Reader lock
- Upgrade
- Writer lock

Waiting: Reader lock is released
## 32 threads

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

### 9.2.0

**Tasks:** 116, **544 thr:** 18 running
**Load average:** 5.71 3.22 1.65
**Uptime:** 62 days, 21:18:56

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

### PoC

**Tasks:** 119, **544 thr:** 33 running
**Load average:** 17.08 13.70 6.76
**Uptime:** 62 days, 21:36:14

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |

---

- **32 threads**
- **9.2.0**
- **PoC**