**Mongo DB Scaling report**

1. Thread scaling baseline
2. Emon output snapshot
3. Perf output snapshot with 30 threads and 80 threads.
4. Mongostat per run
5. Mongotop per run
6. Perf stat for the schedule, cpu migrations
7. Thread affinity code change to reduce migration,
   1. Migration improvement
   2. Performance improvement
8. **WIP: to track calls to a spin lock in a kernel from the user function.**

**Thread scaling baseline**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Raw Throughput | | | | | | | | | |
| Threads | 1 | 2 | 4 | 8 | 10 | 20 | 30 | 40 | 80 |
| Throughput | 19,376.50 | 38,394.79 | 76,279.39 | 145,476.76 | 182,177.90 | 309,835.41 | 348,174.86 | 341,791.53 | 308,905.11 |
| IPC | 0.89 | 0.87 | 0.86 | 0.85 | 0.82 | 0.66 | 0.56 | 0.46 | 0.47 |
| Utilization | 175 | 3.42 | 6.69 | 13.21 | 25.91 | 47.54 | 62.99 | 75.31 | 71.27 |

**EMON Data:**

|  |  |  |  |
| --- | --- | --- | --- |
| Summary |  |  |  |
| Threads | 1 | 30 | 80 |
| Throughput | 18,967.00 | 329,545 | 298,414.00 |
| IPC | 0.89 | 0.66 | 0.47 |
| **CPI** | 1.12 | 1.51 | 2.13 |
| cpi\_fe | 0.59 | 0.61 | 0.61 |
| cpi\_bs | 0.03 | 0.03 | 0.04 |
| cpi\_be | 0.29 | 0.28 | 0.30 |
| cpi\_ret | 0.21 | 0.20 | 0.17 |
| metric\_CPU operating frequency (in GHz) | 3.35 | 2.99 | 3.00 |
| metric\_CPU utilization % | 1.74 | 47.54 | 71.27 |
| metric\_CPU utilization% in kernel mode | 0.79 | 18.35 | 34.10 |
| metric\_cycles per txn | 245,925.74 | 345,501.46 | 573,157.59 |
| metric\_NUMA %\_Reads addressed to local DRAM | 98.93 | 99.66 | 99.66 |
| metric\_NUMA %\_Reads addressed to remote DRAM | 1.07 | 0.34 | 0.34 |
| metric\_TMA\_Frontend\_Bound(%) | 52.65 | 54.45 | 54.52 |
| metric\_TMA\_....ICache\_Misses(%) | 13.64 | 13.85 | 12.42 |
| metric\_TMA\_....ITLB\_Misses(%) | 12.26 | 13.10 | 16.64 |
| metric\_TMA\_Bad\_Speculation(%) | 2.47 | 3.10 | 3.71 |
| metric\_TMA\_Backend\_Bound(%) | 26.10 | 24.87 | 26.66 |
| metric\_TMA\_..Memory\_Bound(%) | 11.62 | 13.11 | 14.07 |
| metric\_TMA\_......DTLB\_Load(%) | 7.37 | 8.74 | 9.19 |
| metric\_TMA\_......DTLB\_Store(%) | 3.29 | 3.93 | 4.16 |
| metric\_TMA\_..Core\_Bound(%) | 14.48 | 11.76 | 12.59 |
| metric\_TMA\_........Store\_STLB\_Miss(%) | 0.05 | 0.27 | 0.81 |
| metric\_TMA\_Retiring(%) | 18.79 | 17.58 | 15.12 |
| INST\_RETIRED.ANY | 218,753.13 | 229,339.14 | 269,528.56 |
| INST\_RETIRED.ANY:SUP | 75,252.82 | 78,431.62 | 117,886.62 |
| MISC\_RETIRED.PAUSE\_INST | 0.81 | 42.61 | 713.48 |
| UNC\_CHA\_CLOCKTICKS | 5,071,092.82 | 287,730.20 | 308,771.86 |
| UNC\_CHA\_DISTRESS\_ASSERTED.DPT\_LOCAL | 12.07 | 588.12 | 1,641.23 |
| UNC\_CHA\_DISTRESS\_ASSERTED.DPT\_NONLOCAL | 228.15 | 21,881.04 | 63,196.53 |
| UNC\_CHA\_DISTRESS\_ASSERTED.VERT | - | 13.44 | 21.97 |

Some of the CPI related issues (I$, ITLB misses) can and are reduced with code layout optimization using BOLT. Refer <https://jira.devtools.intel.com/browse/RPO-981> for the details.

|  |
| --- |
| Perf report details:  0.20% conn353 mongod [.] \_\_wt\_row\_search  0.20% conn360 mongod [.] \_\_wt\_row\_search  0.20% conn344 mongod [.] \_\_wt\_row\_search  0.20% conn354 mongod [.] \_\_wt\_row\_search  0.20% conn363 mongod [.] \_\_wt\_row\_search  0.20% conn366 mongod [.] \_\_wt\_row\_search  0.20% conn342 mongod [.] \_\_wt\_row\_search  0.20% conn350 mongod [.] \_\_wt\_row\_search  0.20% conn345 mongod [.] \_\_wt\_row\_search  0.19% conn365 mongod [.] \_\_wt\_row\_search  0.19% conn339 mongod [.] \_\_wt\_row\_search  0.19% conn355 mongod [.] \_\_wt\_row\_search  0.19% conn348 mongod [.] \_\_wt\_row\_search  0.19% conn357 mongod [.] \_\_wt\_row\_search  0.19% conn349 mongod [.] \_\_wt\_row\_search  0.19% conn352 mongod [.] \_\_wt\_row\_search  0.19% conn338 mongod [.] \_\_wt\_row\_search  0.19% conn347 mongod [.] \_\_wt\_row\_search  0.19% conn361 mongod [.] \_\_wt\_row\_search  0.19% conn362 mongod [.] \_\_wt\_row\_search  0.19% conn356 mongod [.] \_\_wt\_row\_search  0.19% conn343 mongod [.] \_\_wt\_row\_search  0.19% conn340 mongod [.] \_\_wt\_row\_search  0.19% conn341 mongod [.] \_\_wt\_row\_search  0.19% conn364 mongod [.] \_\_wt\_row\_search  0.19% conn337 mongod [.] \_\_wt\_row\_search  0.19% conn351 mongod [.] \_\_wt\_row\_search  0.19% conn359 mongod [.] \_\_wt\_row\_search  0.19% conn346 mongod [.] \_\_wt\_row\_search  0.19% conn358 mongod [.] \_\_wt\_row\_search  …  …. |

|  |
| --- |
| #Threads: 30  # Samples: 1M of event 'cycles'  Event count (approx.): 3357435133299  % Cycles Object  64.99% mongod  22.81% [kernel.vmlinux]  5.27% libpthread-2.31.so  3.48% libc-2.31.so  0.89% [nf\_conntrack]  0.64% [bridge]  0.57% [br\_netfilter]  0.39% [unknown]  0.31% [vdso]  0.24% [ip\_tables]  0.16% [veth]  0.11% [nf\_nat]  0.09% [xt\_conntrack]  0.03% [nf\_defrag\_ipv4]  0.02% [iptable\_filter]  0.00% [overlay]  0.00% [ixgbe]  0.00% [kvm] |

Perf output snapshot (30 seconds output)

|  |
| --- |
| **One of the thread (conn353) perf data view**  0.20% conn353 mongod [.] \_\_wt\_row\_search  0.05% conn353 mongod [.] operator new[]  0.05% conn353 libpthread-2.31.so [.] pthread\_attr\_getstacksize  0.04% conn353 libpthread-2.31.so [.] pthread\_setschedparam  0.04% conn353 libpthread-2.31.so [.] \_\_pthread\_mutex\_trylock  0.04% conn353 mongod [.] \_\_config\_next  0.04% conn353 mongod [.] mongo::latch\_detail::Mutex::\_onUnlock  0.04% conn353 mongod [.] mongo::latch\_detail::Mutex::\_onQuickLock ▒  0.03% conn353 libc-2.31.so [.] \_\_strlen\_avx2  0.03% conn353 [kernel.vmlinux] [k] try\_to\_wake\_up  0.03% conn353 mongod [.] std::\_Sp\_counted\_base<(\_\_gnu\_cxx::\_Lock\_polic  0.03% conn353 mongod [.] \_\_wt\_page\_in\_func  0.02% conn353 mongod [.] tc\_deletearray\_aligned\_nothrow  0.02% conn353 [kernel.vmlinux] [k] psi\_task\_change  0.02% conn353 libc-2.31.so [.] \_\_strcasecmp\_l\_avx  0.02% conn353 mongod [.] \_\_wt\_cursor\_cache\_get  0.02% conn353 mongod [.] mongo::(anonymous namespace)::DiagnosticListe  0.02% conn353 mongod [.] mongo::latch\_detail::Mutex::\_onContendedLock  0.02% conn353 mongod [.] mongo::(anonymous namespace)::DiagnosticListe  0.02% conn353 [kernel.vmlinux] [k] tcp\_recvmsg  0.02% conn353 mongod [.] mongo::StringData::StringData  0.02% conn353 mongod [.] mongo::CollectionCatalog::get  0.02% conn353 mongod [.] mongo::runQuery |

* Total clocks (from all 30 threads) in the “wt\_row\_search” function is approx. 1.49%
* Total clocks in native\_queued\_spin\_lock() function on the behalf of each “conn” thread is approx. 0.6%
* Total clocks in mongo::(anonymous namespace)::DiagnosticListener::onQuickLock from various “conn” thread is approx. 1.5%

Let’s see next table with 80 threads.

|  |
| --- |
| # Samples: 9M of event 'cycles'  # Event count (approx.): 5152842369888  #  # Overhead Command Shared Object Symbol  5.34% swapper [kernel.vmlinux] [k] acpi\_processor\_ffh\_cstate\_enter  0.37% swapper [kernel.vmlinux] [k] set\_next\_entity  0.34% swapper [kernel.vmlinux] [k] \_\_sched\_text\_start  0.26% swapper [kernel.vmlinux] [k] do\_idle  0.25% swapper [kernel.vmlinux] [k] menu\_select  0.20% swapper [kernel.vmlinux] [k] switch\_mm\_irqs\_off  0.18% conn244 [kernel.vmlinux] [k] native\_queued\_spin\_lock\_slowpath  0.17% conn192 [kernel.vmlinux] [k] native\_queued\_spin\_lock\_slowpath  ...  78 calls to native\_queued\_spin\_lock\_slowpath  ...  0.17% conn200 [kernel.vmlinux] [k] native\_queued\_spin\_lock\_slowpath  0.17% swapper [kernel.vmlinux] [k] nr\_iowait\_cpu  0.17% conn203 [kernel.vmlinux] [k] native\_queued\_spin\_lock\_slowpath  0.17% conn175 [kernel.vmlinux] [k] native\_queued\_spin\_lock\_slowpath  ...  0.02% Service.Fixed-0 mongod [.] asio::detail::epoll\_reactor::run  0.01% conn245 libpthread-2.31.so [.] \_\_pthread\_mutex\_lock  0.01% conn212 [kernel.vmlinux] [k] psi\_task\_change  0.01% conn242 mongod [.] operator new[]  0.01% conn231 libpthread-2.31.so [.] \_\_pthread\_mutex\_lock  …  0.01% conn215 mongod [.] operator new[]  0.01% conn170 mongod [.] operator new[]  0.01% conn237 libpthread-2.31.so [.] \_\_pthread\_mutex\_lock  0.01% conn227 libpthread-2.31.so [.] \_\_pthread\_mutex\_lock  0.01% conn212 libpthread-2.31.so [.] \_\_pthread\_mutex\_lock  0.01% conn207 mongod [.] operator new[]  0.01% conn218 libpthread-2.31.so [.] \_\_pthread\_mutex\_lock  0.01% conn189 mongod [.] operator new[]  0.01% conn175 mongod [.] operator new[]  0.01% conn199 libpthread-2.31.so [.] \_\_pthread\_mutex\_lock  0.01% conn241 libpthread-2.31.so [.] \_\_pthread\_mutex\_lock  0.01% conn227 mongod [.] operator new[]  0.01% conn194 [kernel.vmlinux] [k] psi\_task\_change  0.01% conn244 mongod [.] operator new[]  0.01% conn189 libpthread-2.31.so [.] \_\_pthread\_mutex\_lock  0.01% conn218 mongod [.] operator new[]  0.01% conn169 libpthread-2.31.so [.] \_\_pthread\_mutex\_lock  0.01% conn178 libpthread-2.31.so [.] \_\_pthread\_mutex\_lock  0.01% conn216 mongod [.] operator new[]  0.01% conn177 libpthread-2.31.so [.] \_\_pthread\_mutex\_lock  … |

As you can see from this table that most mongodb threads are holding a “mutex” lock.

According to the perf data,

* Total clocks (from all 80 threads) in the “wt\_row\_search” function **is approximately 0.8%**
* Total clocks in native\_queued\_spin\_lock() function on the behalf of each “conn” thread is **approximately 13.6%**
* Total clocks in mongo::(anonymous namespace)::DiagnosticListener::onQuickLock from various “conn” thread is **approximately 1.3%**
* Perf data also shows that the total clocks per “conn” thread is **approximately 0.38%** (including time in all types of locks)

**Perf lock and false sharing investigation:**

**Perf stat for the schedule, cpu migrations data point:**

$sudo perf stat -e "sched:sched\_process\*,task:\*,L1-dcache-loads,L1-dcache-load-misses,cycles,cs,faults,migrations" -d -d -d -a -- sleep 60

|  |  |  |
| --- | --- | --- |
|  | Baseline | Baseline |
| Threads | 40 | 80 |
| Throughput | 337,008.97 | 304,580.89 |
| sched:sched\_process\_free | 13 | 18 |
| sched:sched\_process\_exit | 13 | 16 |
| sched:sched\_process\_wait | 34 | 34 |
| sched:sched\_process\_fork | 11 | 15 |
| sched:sched\_process\_exec | 10 | 11 |
| sched:sched\_process\_hang | 0 | 0 |
| task:task\_newtask | 11 | 15 |
| task:task\_rename | 10 | 14 |
| L1-dcache-loads | 1,406,341,217,349 | 1,321,016,036,512 |
| L1-dcache-load-misses | 140,998,950,663 | 148,306,686,642 |
| cycles | 8,938,835,361,677 | 9,556,957,123,623 |
| cs | 199,014,407 | 179,466,722 |
| faults | 3,908,342 | 2,619,539 |
| migrations | **411,085** | **37,726,014** |
| L1-dcache-loads | 1,405,541,600,119 | 1,321,088,938,840 |
| L1-dcache-load-misses | 140,852,259,993 | 148,349,495,168 |
| LLC-loads | 27,442,725,840 | 39,314,612,447 |
| LLC-load-misses | 97182184700.00% | 1,019,494,242 |
| L1-icache-load-misses | 514,863,336,537 | 500,997,125,592 |
| dTLB-loads | 1,405,809,181,313 | 1,321,617,878,195 |
| dTLB-load-misses | 444985364000.00% | 8,381,345,576 |
| iTLB-load-misses | 8,191,985,418 | 11,331,736,113 |
| Elapsed Time (sec) | 60 | 60 |

As you can see from the data, the CPU migrations have been increased with increased threads.

Following code is added to ping each thread to the cpu.

diff --git a/src/mongo/transport/service\_executor\_utils.cpp b/src/mongo/transport/service\_executor\_utils.cpp

index 13cba5d994d..119133fd99a 100644

--- a/src/mongo/transport/service\_executor\_utils.cpp

+++ b/src/mongo/transport/service\_executor\_utils.cpp

@@ -44,6 +44,7 @@

#include "mongo/util/debug\_util.h"

#include "mongo/util/scopeguard.h"

#include "mongo/util/thread\_safety\_context.h"

+#include <sched.h>

#if !defined(\_WIN32)

#include <sys/resource.h>

@@ -59,6 +60,16 @@ namespace mongo {

namespace {

void\* runFunc(void\* ctx) {

+ pthread\_t tid = pthread\_self();

+ int cpuid = sched\_getcpu();

+ fprintf(stdout, "------> thread: %ld on cpu: %d\n", tid, cpuid);

+

+ cpu\_set\_t cpuset;

+ CPU\_ZERO(&cpuset); //clears the cpuset

+ CPU\_SET(cpuid, &cpuset); //set CPU cpuid in cpuset

+ sched\_setaffinity(0, sizeof(cpuset), &cpuset);

+

+

auto taskPtr =

std::unique\_ptr<unique\_function<void()>>(static\_cast<unique\_function<void()>\*>(ctx));

(\*taskPtr)();

With this code change, I was able to see big **reduction (> 70%) in the “migrations”** count and approximately **2% to 5% throughput improvement** for 40 and 80 threaded run respectively.

Drop in the migrations

|  |  |  |
| --- | --- | --- |
|  | Thread-affinity | Thread-affinity |
|  | 40 | 80 |
| Throughput | 342,339.13 | 325,860.27 |
| sched:sched\_process\_free | 40 | 43 |
| sched:sched\_process\_exit | 39 | 42 |
| sched:sched\_process\_wait | 94 | 109 |
| sched:sched\_process\_fork | 35 | 40 |
| sched:sched\_process\_exec | 31 | 36 |
| sched:sched\_process\_hang | 0 | 0 |
| task:task\_newtask | 35 | 40 |
| task:task\_rename | 34 | 39 |
| L1-dcache-loads | 1,371,478,261,764 | 1,397,652,766,286 |
| L1-dcache-load-misses | 134,375,048,259 | 148,352,035,975 |
| cycles | 8,035,705,103,230 | 11,636,054,095,013 |
| cs | 164,336,498 | 178,628,471 |
| faults | 4,188,202 | 3,993,059 |
| migrations | **38,418** | **9,116,093** |
| L1-dcache-loads | 1,371,011,714,413 | 1,396,817,411,470 |
| L1-dcache-load-misses | 134,294,621,329 | 148,244,429,483 |
| LLC-loads | 27,550,039,711 | 32,720,502,471 |
| LLC-load-misses | 1,049,323,298 | 1,035,648,981 |
| L1-icache-load-misses | 535,918,102,076 | 491,637,263,588 |
| dTLB-loads | 1,370,459,278,347 | 1,397,851,345,369 |
| dTLB-load-misses | 3,434,140,800 | 6,692,012,412 |
| iTLB-load-misses | 6,407,918,838 | 9,886,306,215 |
| Elapsed Time (sec) | 60 | 60 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Raw Throughput with Thread-affinity change | | | | | | | | | |
| Threads | 1 | 2 | 4 | 8 | 10 | 20 | 30 | 40 | 80 |
| Throughput | 20,030 | 38,220 | 76,082 | 143,931 | 174,885 | 305,840 | 344,872 | 348,631 | 325,860 |

Overall performance impact with this change,



**4% drop**

Data from mongoto and mongostat,

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| $ mongotop 20 | |  |  |  |
|  | Threads | total (ms) | read (ms) | write (ms) |
| ycsb.usertable | 1 | 3,938 | 3,938 | 0 |
|  | 2 | 15,855 | 15,855 | 0 |
|  | 4 | 31,292 | 31,292 | 0 |
|  | 8 | 62,090 | 62,090 | 0 |
|  | 10 | 78,074 | 78,074 | 0 |
|  | 20 | 155,448 | 155,448 | 0 |
|  | 30 | 221,053 | 221,053 | 0 |
|  | 40 | 271,381 | 271,381 | 0 |
|  | 80 | 272,245 | 272,245 | 0 |

Note: The table shows the amount of time that this mongod spent performing read operations during each interval (20 seconds in above case)

$ mongostat 20 (polling every 20 seconds)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | Mem pages | |  |  |  |  |  |  |  |  |
| Thrs | insert | query | update | delete | dirty | used | flushes | vsize | res | qrw | arw | net\_in (m) | net\_out (m) | conn |
| 1 | 0 | 19,741 | \*0 | \*0 | 0.00% | 9.90% | 1 | 15.0G | 12.0G | 0 | 0 | 1.6 | 23.8 | 4 |
| 2 |  | 38,309 |  |  | 0.00% | 9.90% |  |  |  | 0 | 0 | 3.1 | 46.1 | 5 |
| 4 |  | 75,441 |  |  | 0.00% | 9.90% |  |  |  | 0 | 1 | 6.08 | 90.5 | 7 |
| 8 |  | 146,538 |  |  | 0.00% | 9.90% |  |  |  | 0 | 1 | 11.8 | 176 | 11 |
| 10 |  | 179,606 |  |  | 0.00% | 9.90% |  |  |  | 2 | 2 | 14.5 | 216 | 13 |
| 20 |  | 305,713 |  |  | 0.00% | 9.90% |  |  |  | 5 | 6 | 24.7 | 368 | 23 |
| 30 |  | 342,416 |  |  | 0.00% | 9.90% |  |  |  | 0 | 10 | 27.7 | 412 | 33 |
| 40 |  | 340,204 |  |  | 0.00% | 9.90% |  |  |  | 4 | 10 | 27.5 | 410 | 43 |
| 80 |  | 307,483 |  |  | 0.00% | 9.90% |  |  |  | 3 | 16 | 24.8 | 370 | 83 |

Note: **qrw** The, length of the queue of clients waiting to read data from the MongoDB instance.

**arw** The, number of active clients performing read operations.

**net\_in** The, amount of network traffic, in bytes, received by the MongoDB instance

**net\_out** The amount of network traffic, in bytes, sent by the MongoDB instance.