POSTGRESQL
AT LOW LEVEL

STAY CURIOUS!

DMITRY DOLGOV

07-03-2019
patroni & postgres-operator
pg_stat_*
pg_stat_*
CPU/IO
???

CG
OS
PG
All right, then. Keep your secrets
Plan?
A bit chaotic
Info sources

source code
strace/GDB/Perf
procfs/sysfs
BPF/eBPF/BCC
Shared memory

ERROR: could not resize shared memory segment "/PostgreSQL.699663942" to 50438144 bytes:
   No space left on device
# strace -k -p PID
openat(AT_FDCWD, "/dev/shm/PostgreSQL.62223175"
ftruncate(176, 50438144) = 0
fallocate(176, 0, 0, 50438144) = -1 ENOSPC
  > libc-2.27.so(posix_fallocate+0x16) [0x114f76]
  > postgres(dsm_create+0x67) [0x377067]
      ...
  > postgres(ExecInitParallelPlan+0x360) [0x254a80]
  > postgres(ExecGather+0x495) [0x269115]
  > postgres(standard_ExecutorRun+0xfd) [0x25099d]
      ...
  > postgres(exec_simple_query+0x19f) [0x39afdf]
vDSO

# strace -k -p PID on XEN
gettimeofday({tv_sec=1550586520, tv_usec=313499}, NULL) = 0
> [vdso]() [0xef0]

Two frequently used system calls are 77% slower on AWS EC2
# Experiment 1

transaction type: pg_long.sql

latency average = 1312.903 ms

# Experiment 2

SQL script 1: pg_long.sql
- weight: 1 (targets 50.0% of total)
- latency average = 1426.928 ms

SQL script 2: pg_short.sql
- weight: 1 (targets 50.0% of total)
- latency average = 303.092 ms
Scheduling
Scheduling

T2

T3
Scheduling

T2

T3

C

C
# perf record -e cache-misses,cpu-migrations

# Experiment 1
12,396,382,649 cache-misses # 28.562%
2,750 cpu-migrations

# Experiment 2
20,665,817,234 cache-misses # 28.533%
10,460 cpu-migrations
<table>
<thead>
<tr>
<th>time</th>
<th>cpu</th>
<th>01234</th>
<th>task name [tid/pid]</th>
<th>wait time (msec)</th>
<th>sch delay (msec)</th>
<th>run time (msec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4227.834476 [0003]</td>
<td>s</td>
<td>postgres[12935]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>4227.834895 [0003]</td>
<td>s</td>
<td>postgres[12935]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.418</td>
<td></td>
</tr>
<tr>
<td>4227.835478 [0003]</td>
<td>s</td>
<td>postgres[25080]</td>
<td>0.000</td>
<td>0.040</td>
<td>0.583</td>
<td></td>
</tr>
<tr>
<td>4227.836485 [0003]</td>
<td>s</td>
<td>postgres[25080]</td>
<td>0.000</td>
<td>0.000</td>
<td>1.007</td>
<td></td>
</tr>
<tr>
<td>4227.837402 [0003]</td>
<td>s</td>
<td>postgres[25080]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.996</td>
<td></td>
</tr>
<tr>
<td>4227.837784 [0003]</td>
<td>s</td>
<td>postgres[25080]</td>
<td>0.000</td>
<td>0.000</td>
<td>0.302</td>
<td></td>
</tr>
<tr>
<td>4227.837989 [0003]</td>
<td>m</td>
<td>postgres[25080]</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
migrated: 25077 cpu 1 -> 3
| 4227.837993 [0003] | s   | postgres[25080]   | 0.000              | 0.000            | 0.208           |
| 4227.848487 [0003] | s   | postgres[25080]   | 0.000              | 0.000            | 10.493          |
| 4227.848991 [0003] | s   | postgres[25080]   | 0.000              | 0.000            | 0.504           |
| 4227.849487 [0003] | s   | postgres[25080]   | 0.000              | 0.000            | 0.495           |
| 4227.849748 [0003] | s   | postgres[25080]   | 0.000              | 0.000            | 0.260           |
| 4227.849912 [0003] | s   | postgres[25080]   | 0.000              | 0.000            | 0.164           |
| 4227.851477 [0001] | s   | postgres[25082]   | 0.000              | 0.000            | 0.000           |
| 4227.851481 [0002] | s   | postgres[25080]   | 0.000              | 0.000            | 0.000           |
| 4227.851778 [0003] | s   | postgres[12935]   | 15.017             | 0.000            | 1.866           |
migrated: postgres[25083] cpu 1 -> 3
| 4227.852259 [0003] | m   | postgres[12935]   |                   |                  |                 |
| 4227.852263 [0003] | s   | postgres[12935]   | 0.000              | 0.000            | 0.484           |
| 4227.852477 [0003] | s   | postgres[25083]   | 0.000              | 0.058            | 0.214           |
| 4227.852478 [0001] | s   | postgres[25082]   | 0.000              | 0.000            | 1.001           |
| 4227.852614 [0002] | s   | postgres[12935]   | 0.000              | 0.000            | 1.133           |
Huge pages

transparent vs classic
TLB misses are faster and less frequent
Huge pages

# perf record -e dTLB-loads,dTLB-stores -p PID
# huge_pages on
Samples: 832K of event 'dTLB-load-misses'
Event count (approx.): 640614445 : ~19% less
Samples: 736K of event 'dTLB-store-misses'
Event count (approx.): 72447300 : ~29% less

# huge_pages off
Samples: 894K of event 'dTLB-load-misses'
Event count (approx.): 784439650
Samples: 822K of event 'dTLB-store-misses'
Event count (approx.): 101471557
VM

- Lock holder preemption problem
- Lock waiter preemption problem
- Intel PLE (pause loop exiting)
- PLE_Gap, PLE_Window

Intel® 64 and IA-32 Architectures Software Developer’s Manual, Vol. 3
vCPU

vC1  vC2  vC3  vC4

Hypervisor
vCPU

Hypervisor

vC1  vC2  vC3  vC4
# latency average = 17.782 ms
=> modprobe kvm-intel ple_gap=128
=> perf record -e kvm:kvm_exit
reason PAUSE_INSTRUCTION 306795

# latency average = 16.858 ms
=> modprobe kvm-intel ple_gap=0
=> perf record -e kvm:kvm_exit
reason PAUSE_INSTRUCTION 0
And now for something completely different
Tunables

# from /proc/sys/kernel/
sched_wakeup_granularity_ns
# default = 1 msec * (1 + ilog(ncpus))
Userspace

Bytecode
Userspace

Bytecode

Regs

Stack

Maps
# pgbench and pg_dump

<table>
<thead>
<tr>
<th>usecs</th>
<th>count</th>
<th>distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 -&gt; 1</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td>2 -&gt; 3</td>
<td>4604</td>
<td>**</td>
</tr>
<tr>
<td>4 -&gt; 7</td>
<td>6812</td>
<td>****</td>
</tr>
<tr>
<td>8 -&gt; 15</td>
<td>14888</td>
<td>***********</td>
</tr>
<tr>
<td>16 -&gt; 31</td>
<td>19267</td>
<td>**********</td>
</tr>
<tr>
<td>32 -&gt; 63</td>
<td>65795</td>
<td>************</td>
</tr>
<tr>
<td>64 -&gt; 127</td>
<td>50454</td>
<td>*******************</td>
</tr>
<tr>
<td>128 -&gt; 255</td>
<td>16393</td>
<td>************</td>
</tr>
<tr>
<td>256 -&gt; 511</td>
<td>5981</td>
<td>***</td>
</tr>
<tr>
<td>512 -&gt; 1023</td>
<td>12300</td>
<td>**********</td>
</tr>
<tr>
<td>1024 -&gt; 2047</td>
<td>48</td>
<td></td>
</tr>
<tr>
<td>2048 -&gt; 4095</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
pgbench and pg_dump

real  1m32.030s
user  1m8.559s
sys   0m1.641s

<table>
<thead>
<tr>
<th>usecs</th>
<th>: count</th>
<th>distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>46</td>
<td>*</td>
</tr>
<tr>
<td>16</td>
<td>189</td>
<td>********</td>
</tr>
<tr>
<td>32</td>
<td>119</td>
<td>****</td>
</tr>
<tr>
<td>64</td>
<td>96</td>
<td>***</td>
</tr>
<tr>
<td>128</td>
<td>93</td>
<td>***</td>
</tr>
<tr>
<td>256</td>
<td>238</td>
<td>**********</td>
</tr>
<tr>
<td>512</td>
<td>323</td>
<td>***********</td>
</tr>
<tr>
<td>1024</td>
<td>1012</td>
<td>*******************************</td>
</tr>
<tr>
<td>2048</td>
<td>47</td>
<td>*</td>
</tr>
</tbody>
</table>
github.com/iovisor/bcc/
github.com/erthalion/postgres-bcc
Cache

Cache contention
Intel RDT
Class of service
Cache

=> llcache_per_query.py bin/postgres

<table>
<thead>
<tr>
<th>PID</th>
<th>QUERY</th>
<th>CPU</th>
<th>REFERENCE</th>
<th>MISS</th>
<th>HIT%</th>
</tr>
</thead>
<tbody>
<tr>
<td>9720</td>
<td>UPDATE pgbench_tellers ... 0</td>
<td>2000</td>
<td>1000</td>
<td>50.00%</td>
<td></td>
</tr>
<tr>
<td>9720</td>
<td>SELECT abalance FROM ... 2</td>
<td>2000</td>
<td>100</td>
<td>95.00%</td>
<td></td>
</tr>
</tbody>
</table>

... 

Total References: 3303100  Total Misses: 599100  Hit Rate: 81.86%
Shared memory

ERROR: could not resize shared memory segment
"/PostgreSQL.699663942" to 504,3144 bytes:
No space left on device

Remember?
Shared memory

=> shmem.py bin/postgres

mmap:
[20439]: 142M
anon shm:
[20439]: 56B
shm:
[postmaster.opts]: 0B
[PostgreSQL.57332071]: 7K
Pages written, kernel
Dirty pages

bgw

linux

chkp

OS Cache

Storage
Dirty pages

bgw

linux

chkp

OS Cache

Storage
Dirty pages

bgw

linux

chkp

OS Cache

Storage
Dirty pages

bgw

linux

chkp

DS Cache

Storage
Writeback (cgroup v1)

/* vmscan.c */
/* The normal page dirty throttling mechanism
 * in balance_dirty_pages() is completely broken
 * with the legacy memcg and direct stalling in
 * shrink_page_list() is used for throttling instead,
 * which lacks all the niceties such as fairness,
 * adaptive pausing, bandwidth proportional
 * allocation and configurability.
 */

static bool sane_reclaim(struct scan_control *sc)
Writeback

=> perf record -e writeback:writeback_written

kworker/u8:1 reason=periodic  nr_pages=101429
kworker/u8:1 reason=background  nr_pages=MAX_ULONG
kworker/u8:3 reason=periodic  nr_pages=101457
Writeback

# pgbench insert workload
=> io_timeouts.py bin/postgres

[18335] END: MAX_SCHEDULE_TIMEOUT
[18333] END: MAX_SCHEDULE_TIMEOUT
[18331] END: MAX_SCHEDULE_TIMEOUT
[18318] truncate pgbench_history: MAX_SCHEDULE_TIMEOUT
Kubernetes

resources:
  requests:
    memory: "64Mi"
    cpu: "250m"
  limits:
    memory: "128Mi"
    cpu: "500m"
Kubernetes

resources:
  requests:
    memory: "64Mi"
    cpu: "250m"
  limits:
    memory: "128Mi"
    cpu: "500m"
Evil ha-ha-ha
Kubernetes

resources:
  requests:
    memory: "64Mi"
    cpu: "250m"
  limits:
    memory: "128Mi"
    cpu: "500m"
Memory reclaim

# only under the memory pressure
=> page_reclaim.py --container 89c33bb3133f

[7382] postgres: 928K
[7138] postgres: 152K
[7136] postgres: 180K
[7468] postgres: 72M
[7464] postgres: 57M
[5451] postgres: 1M
# How to run?

```
# bcc + postgres-bcc

CONFIG_BPF=y
CONFIG_BPF_SYSCALL=y
CONFIG_NET_CLS_BPF=m
CONFIG_NET_ACT_BPF=m
CONFIG_BPF_JIT=y
CONFIG_BPF_EVENTS=y

debugfs on /sys/kernel/debug type debugfs (rw)
```
How to run: container?

# sometimes you also need to let perf know
# where to find debugging symbols, e.g. copy
# from /usr/lib/.debug/
docker run
    --privileged
    --net=container:<container-id>
    --ipc=container:<container-id>
How to run: K8S?

```yaml
spec:
  serviceAccountName: "bcc"
  hostPID: true
  containers:
    - name: "bcc"
      securityContext:
        privileged: true

# 4 * 65536 + 14 * 256 + 96
=> export BCC_LINUX_VERSION_CODE 265824
```
How to break?

# unsafe access

=> perf probe -x bin/postgres --funcs

=> perf probe -x bin/postgres 'ExecCallTriggerFunc trigdata->?'

=> perf record probe_postgres:ExecCallTriggerFunc
How to break?

# non interruptible sleep

=> perf probe -x bin/postgres --funcs

=> perf probe -x bin/postgres 'XLogInsertRecord fpw_lsn'
How to break?

Ubuntu xenial kernel panic in bpf_map_update_elem using ext4slower #1678

Closed stefreak opened this issue on Apr 12, 2018 · 13 comments
Questions?

github.com/erthalion
github.com/erthalion/postgres-bcc
@erthalion
dmitrii.dolgov at zalando dot de
9erthalion6 at gmail dot com