POSTGRESQL
AT LOW LEVEL

STAY CURIOUS!

DMITRY DOLGOV
06-11-2019
patroni & postgres-operator
pg_stat_*
pg_stat_*

CPU/IO

???

CG

OS

PG
pg_stat_*

CPU/IO

???

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

source code
strace/GDB/Perf
procfs/sysfs
BPF/eBPF/BCC
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]
# strace -k -p PID
openat(AT_FDCWD, "/dev/shm/PostgreSQL.62223175"
) = 0
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
Scheduling

T2

C

C

T3
Scheduling

T2

C

T3

C
Andres Freund: New intel MDS vulnerability mitigations cause measurable slowdown
## MDS

<table>
<thead>
<tr>
<th># Children</th>
<th>Self</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>71.06%</td>
<td>0.00%</td>
<td>[.] __libc_start_main</td>
</tr>
<tr>
<td>71.06%</td>
<td>0.00%</td>
<td>[.] PostmasterMain</td>
</tr>
<tr>
<td>56.82%</td>
<td>0.14%</td>
<td>[.] exec_simple_query</td>
</tr>
<tr>
<td>25.19%</td>
<td>0.06%</td>
<td>[k] entry_SYSCALL_64_after_hwframe</td>
</tr>
<tr>
<td>25.14%</td>
<td>0.29%</td>
<td>[k] do_syscall_64</td>
</tr>
<tr>
<td>23.60%</td>
<td>0.14%</td>
<td>[.] standard_ExecutorRun</td>
</tr>
<tr>
<td>#</td>
<td>%</td>
<td>%</td>
</tr>
<tr>
<td>------</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>71.06%</td>
<td>0.00%</td>
<td>[. ]</td>
</tr>
<tr>
<td>71.06%</td>
<td>0.00%</td>
<td>[. ]</td>
</tr>
<tr>
<td>56.82%</td>
<td>0.14%</td>
<td>[. ]</td>
</tr>
<tr>
<td>25.19%</td>
<td>0.06%</td>
<td>[k]</td>
</tr>
<tr>
<td>25.14%</td>
<td>0.29%</td>
<td>[k]</td>
</tr>
<tr>
<td>23.60%</td>
<td>0.14%</td>
<td>[. ]</td>
</tr>
</tbody>
</table>
# Percent | Disassembly of kcore for cycles
# -------- | -----------------------------------------------
0.01% :  nopl 0x0(\%rax,\%rax,1)  
28.94% : verw 0xffe9e1(\%rip)  
0.55% : pop \%rbx  
3.24% : pop \%rbp
## Disassembly of kcore for cycles

<table>
<thead>
<tr>
<th>Percent</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01%</td>
<td><code>nopl 0x0(%rax,%rax,1)</code></td>
</tr>
<tr>
<td>28.94%</td>
<td><code>verw 0xffe9e1(%rip)</code></td>
</tr>
<tr>
<td>0.55%</td>
<td><code>pop %rbx</code></td>
</tr>
<tr>
<td>3.24%</td>
<td><code>pop %rbp</code></td>
</tr>
</tbody>
</table>
MDS

# Overhead   Symbol
# ........   ...................................

25.19%   [k] native_safe_halt
static inline __cpuidle void native_safe_halt(void)
{
    mds_idle_clear_cpu_buffers();
    asm volatile("sti; hlt": : :"memory");
}
static inline __cpuidle void native_safe_halt(void)
{
    mds_idle_clear_cpu_buffers();
    asm volatile("sti; hlt": : :"memory");
}
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
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 = 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
# 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
OS Cache

Storage

WAL

bgw

linux

chkp

OS Cache

Storage
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
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
And now for something completely different
Userspace

Bytecode
Userspace

vfs_read

Bytecode

Regs

...  

Stack

...  

Maps

...
github.com/iovisor/bcc/
github.com/erthalion/postgres-bcc
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></td>
<td>2000</td>
<td>1000</td>
<td>50.00%</td>
</tr>
<tr>
<td>9720</td>
<td>SELECT abalance FROM ... 2</td>
<td></td>
<td>2000</td>
<td>100</td>
<td>95.00%</td>
</tr>
</tbody>
</table>

... 

Total References: **3303100** Total Misses: **599100** Hit Rate: **81.86%**
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"
Kubernetes

resources:

requests:
  memory: "64Mi"
  cpu: "250m"

limits:
  memory: "128Mi"
  cpu: "500m"

soft_limits_in_bytes
limits_in_bytes
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
# 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:
- github.com/erthalion
- github.com/erthalion/postgres-bcc
Twitter:
- @erthalion
Emails:
- dmitrii.dolgov at zalando dot de
- 9erthalion6 at gmail dot com