


zalando

# POSTGRESQL AT LOW LEVEL

STAY CURIOUS!


DMITRY DOLGOV

07-03-2019

patroni & postgres-operator

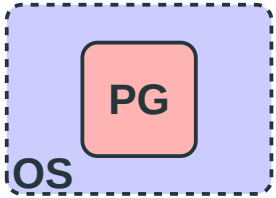
zalando




zalando

pg_stat_*

CPU/IO



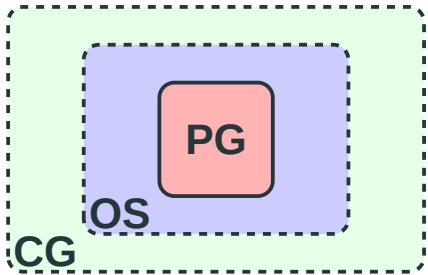



zalando

pg_stat_*

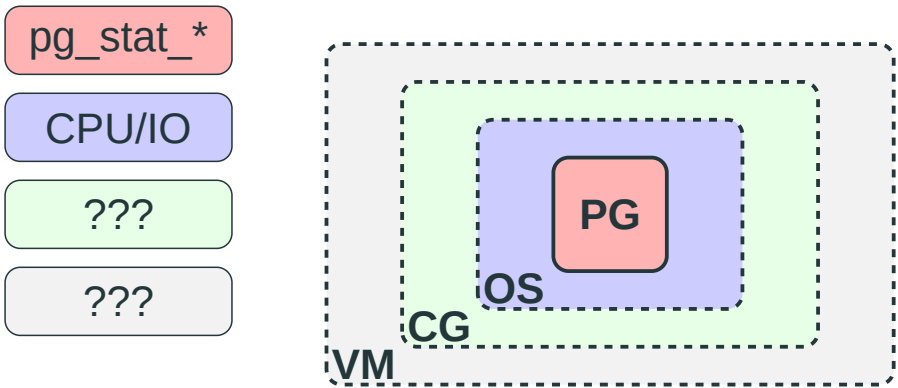
CPU/IO

???



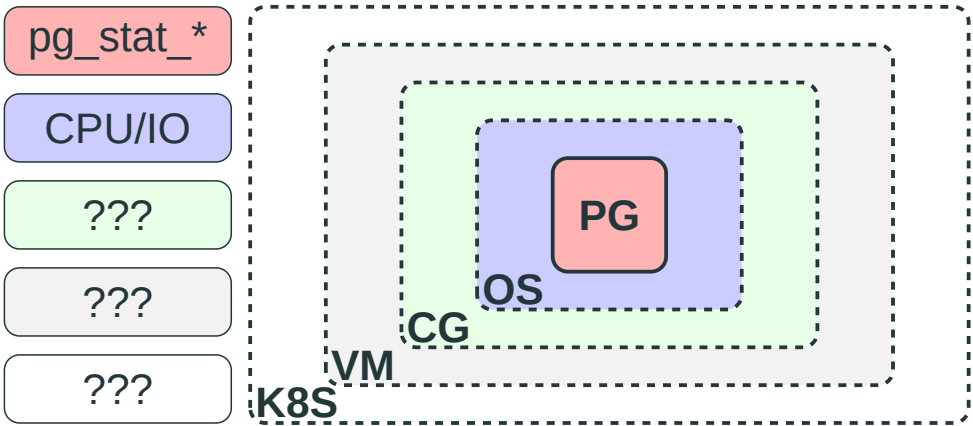



zalando






zalando






zalando




zalando

# Plan?

# A bit chaotic




zalando

## 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]
```

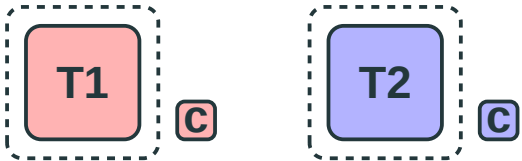
# vDSO

```
# 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
```

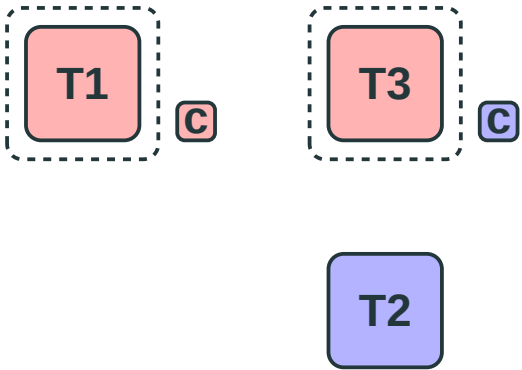
# Scheduling






zalando

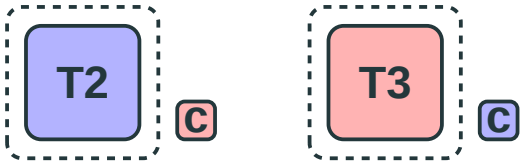
# Scheduling






zalando

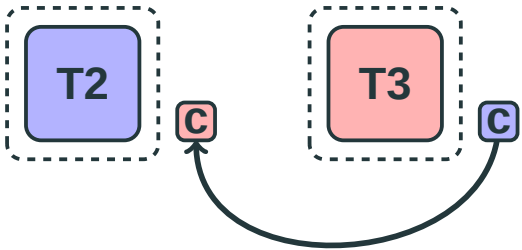
# Scheduling






zalando

# Scheduling






zalando

```
# 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
```

```
         time    cpu  01234  task name                    wait time  sch delay   run time
                             [tid/pid]                       (msec)     (msec)     (msec)
--------------- ------  -----  ---------------------------  ---------  ---------  ---------
   4227.834476 [0003]     s   postgres[12935]                  0.000      0.000      0.000
   4227.834895 [0003]     s   postgres[12935]                  0.000      0.000      0.418
   4227.835478 [0003]     s   postgres[25080]                  0.000      0.040      0.583
   4227.836485 [0003]     s   postgres[25080]                  0.000      0.000      1.007
   4227.837482 [0003]     s   postgres[25080]                  0.000      0.000      0.996
   4227.837784 [0003]     s   postgres[25080]                  0.000      0.000      0.302
   4227.837989 [0003]      m    postgres[25080]                                              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
   4227.852259 [0003]      m    postgres[12935]                                              migrated: postgres[25083] cpu 1 => 3
   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
```



zalando

## Huge pages

transparent vs classic

TLB misses are faster and less frequent

# Huge pages

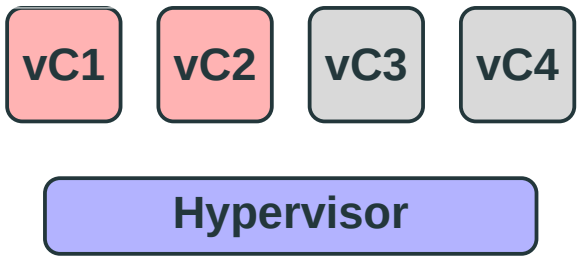
```
# perf record -e dTLB-loads,dTLB-stores -p PID
# huge_pages on
Samples: 832K of event 'dTLB-load-misses'
Event count (approx.): 640614445 : ~19% less
Samples: 736K of event 'dTLB-store-misses'
Event count (approx.): 72447300 : ~29% less

# 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

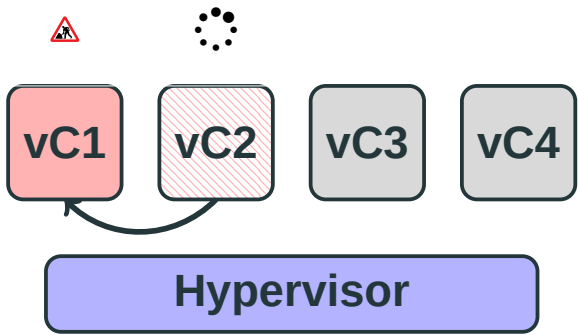
# vCPU






zalando

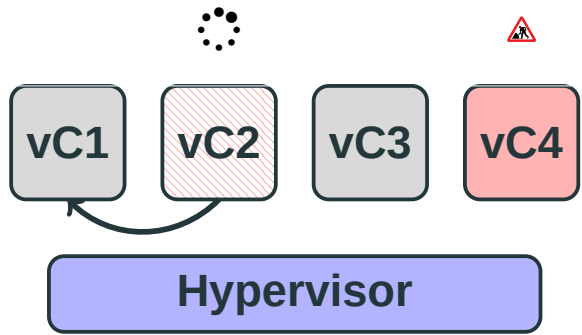
# vCPU






zalando

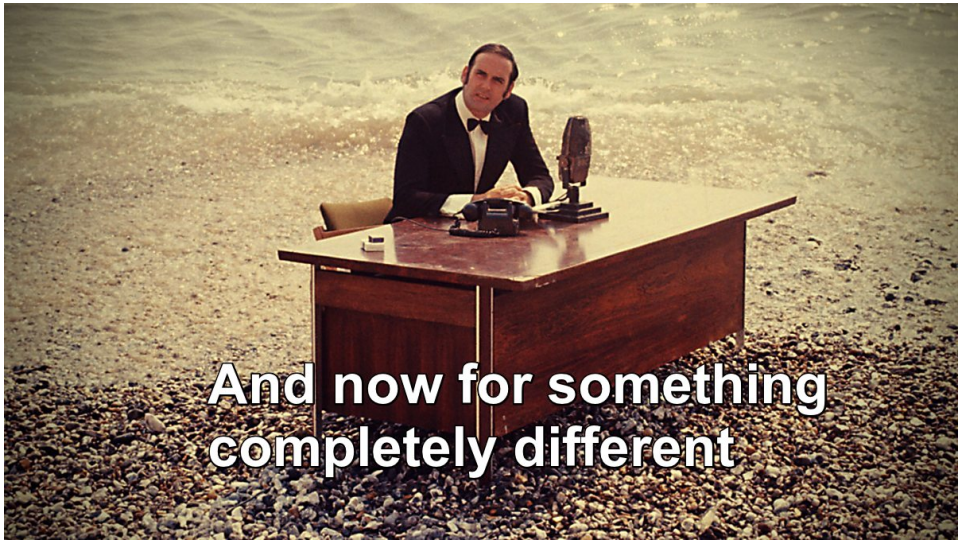
# vCPU






zalando

```
# 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
```






zalando

## Tunables

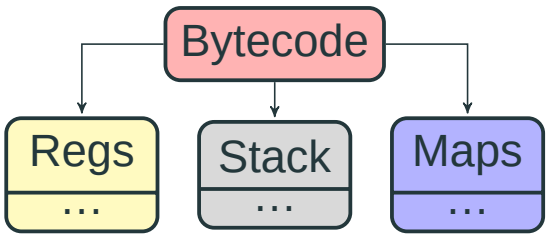
```
# from /proc/sys/kernel/
sched_wakeup_granularity_ns
# default = 1 msec * (1 + ilog(ncpus))
```

# Userspace




zalando

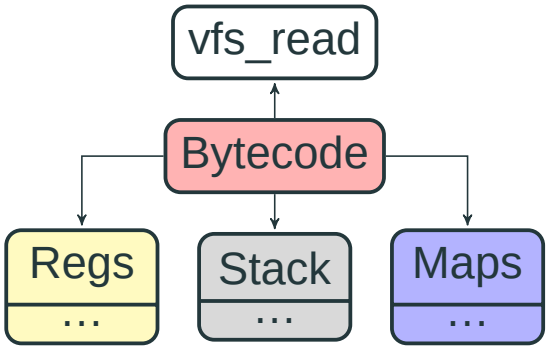
# Userspace






zalando

# Userspace






zalando

# pgbench and pg_dump

```
real    1m38.990s
user    1m9.127s
sys     0m2.066s
```

```
     usecs               : count     distribution
         0 -> 1          : 16       |                                        |
         2 -> 3          : 4604     |**                                      |
         4 -> 7          : 6812     |****                                    |
         8 -> 15         : 14888    |*********                               |
        16 -> 31         : 19267    |***********                             |
        32 -> 63         : 65795    |****************************************|
        64 -> 127        : 50454    |******************************          |
       128 -> 255        : 16393    |*********                               |
       256 -> 511        : 5981     |***                                     |
       512 -> 1023       : 12300    |*******                                 |
      1024 -> 2047       : 48       |                                        |
      2048 -> 4095       : 0        |                                        |
```



zalando

# pgbench and pg_dump

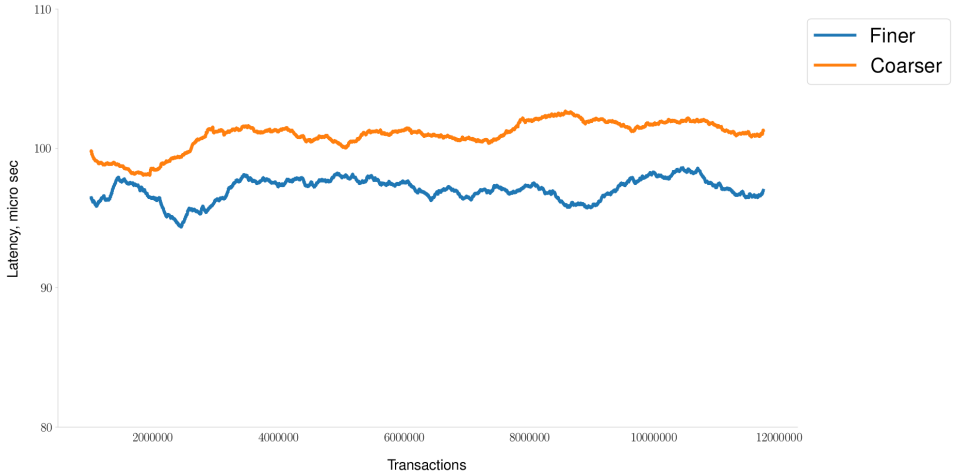
```
real    1m32.030s
user    1m8.559s
sys     0m1.641s
```

```
     usecs               : count     distribution
         0 -> 1          : 1       |                                        |
         2 -> 3          : 8       |                                        |
         4 -> 7          : 25      |                                        |
         8 -> 15         : 46      |*                                       |
        16 -> 31         : 189     |*******                                 |
        32 -> 63         : 119     |****                                    |
        64 -> 127        : 96      |***                                     |
       128 -> 255        : 93      |***                                     |
       256 -> 511        : 238     |*********                               |
       512 -> 1023       : 323     |************                            |
      1024 -> 2047       : 1012    |****************************************|
      2048 -> 4095       : 47      |*                                       |
```



zalando

# Wakeup granularity, microsec






zalando

## Cache

Cache contention

Intel RDT

Class of service

github.com/iovisor/bcc/

github.com/erthalion/postgres-bcc

# Cache

```
=> llcache_per_query.py bin/postgres

PID  QUERY                        CPU REFERENCE MISS   HIT%
9720 UPDATE pgbench_tellers ... 0        2000 1000 50.00%
9720 SELECT abalance FROM   ... 2        2000  100 95.00%
...

Total References: 3303100 Total Misses: 599100 Hit Rate: 81.86%
```

## Shared memory

```
ERROR:  could not resize shared memory segment
"/PostgreSQL.699663942" to 504[illegible]8144 bytes:
    No space left on device
```


zalando

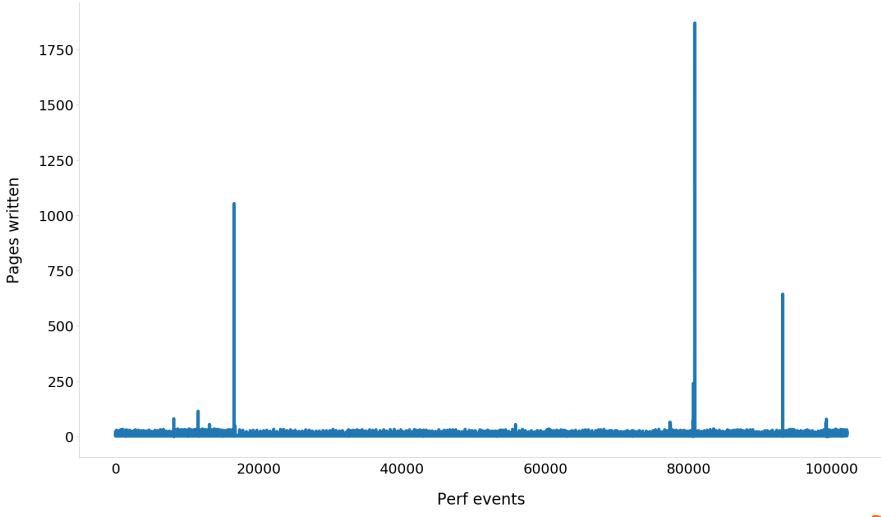


zalando

# Shared memory

```
=> shmem.py bin/postgres

mmap:
[20439]: 142M
anon shm:
[20439]: 56B
shm:
[postmaster.opts]: 0B
[PostgreSQL.57332071]: 7K
```

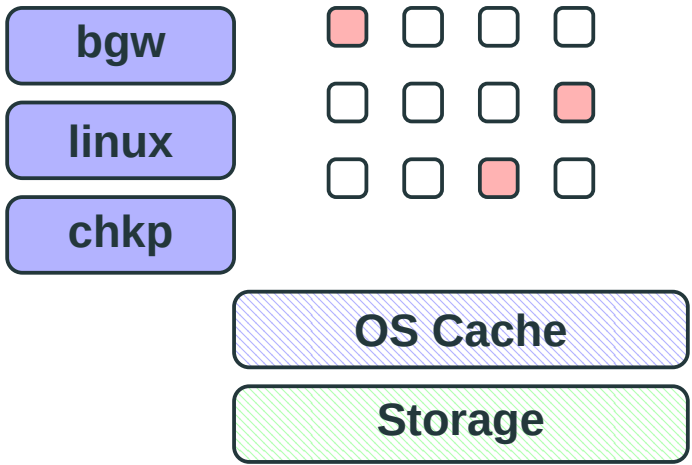
# Pages written, kernel






zalando

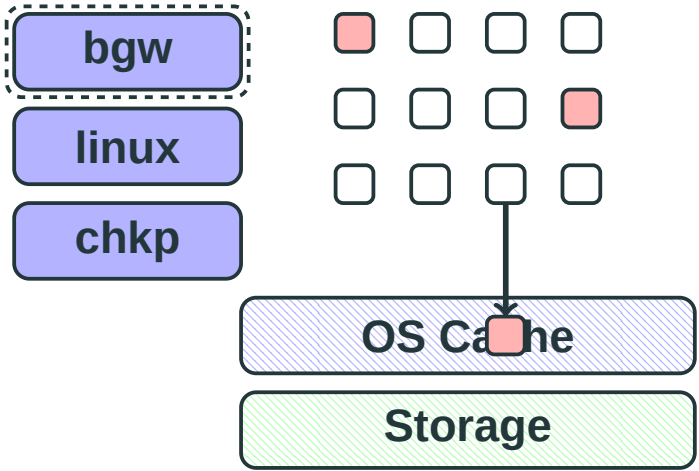
# Dirty pages






zalando

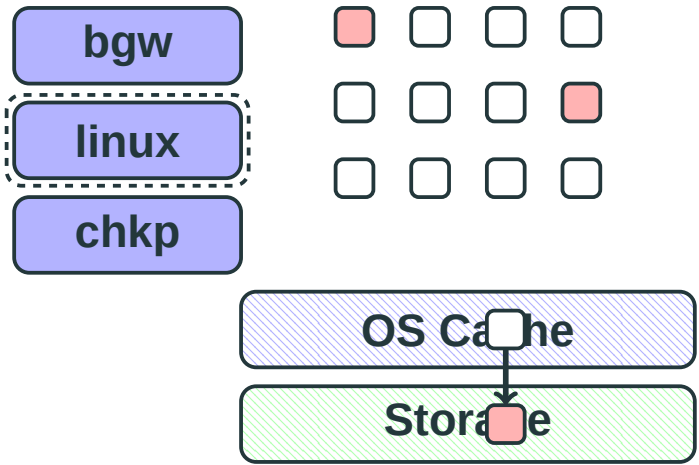
# Dirty pages






zalando

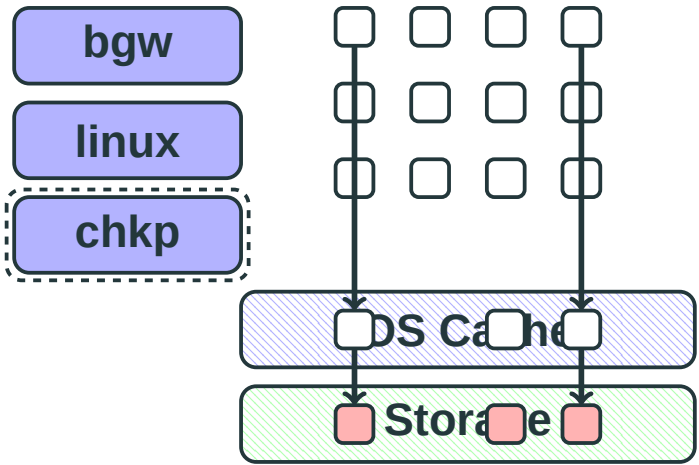
# Dirty pages






zalando

# Dirty pages






zalando

# 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

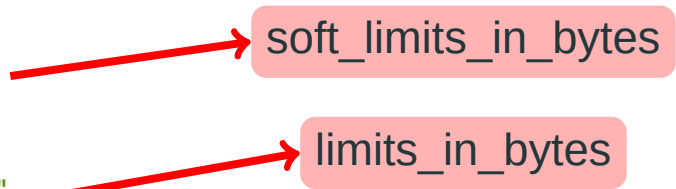
```
# pgbench insert workload
=> io_timeouts.py bin/postgres

[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"
```





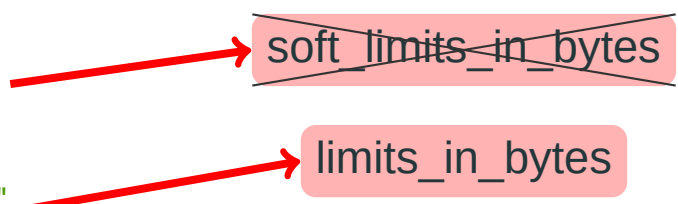

zalando




zalando

# Kubernetes

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






zalando

# 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
    --priviledged
    --net=container:<container-id>
    --ipc=container:<container-id>
```

# How to run: K8S?

```
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?




zalando

## Questions?

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



zalando