



# POSTGRESQL LINUX KERNEL

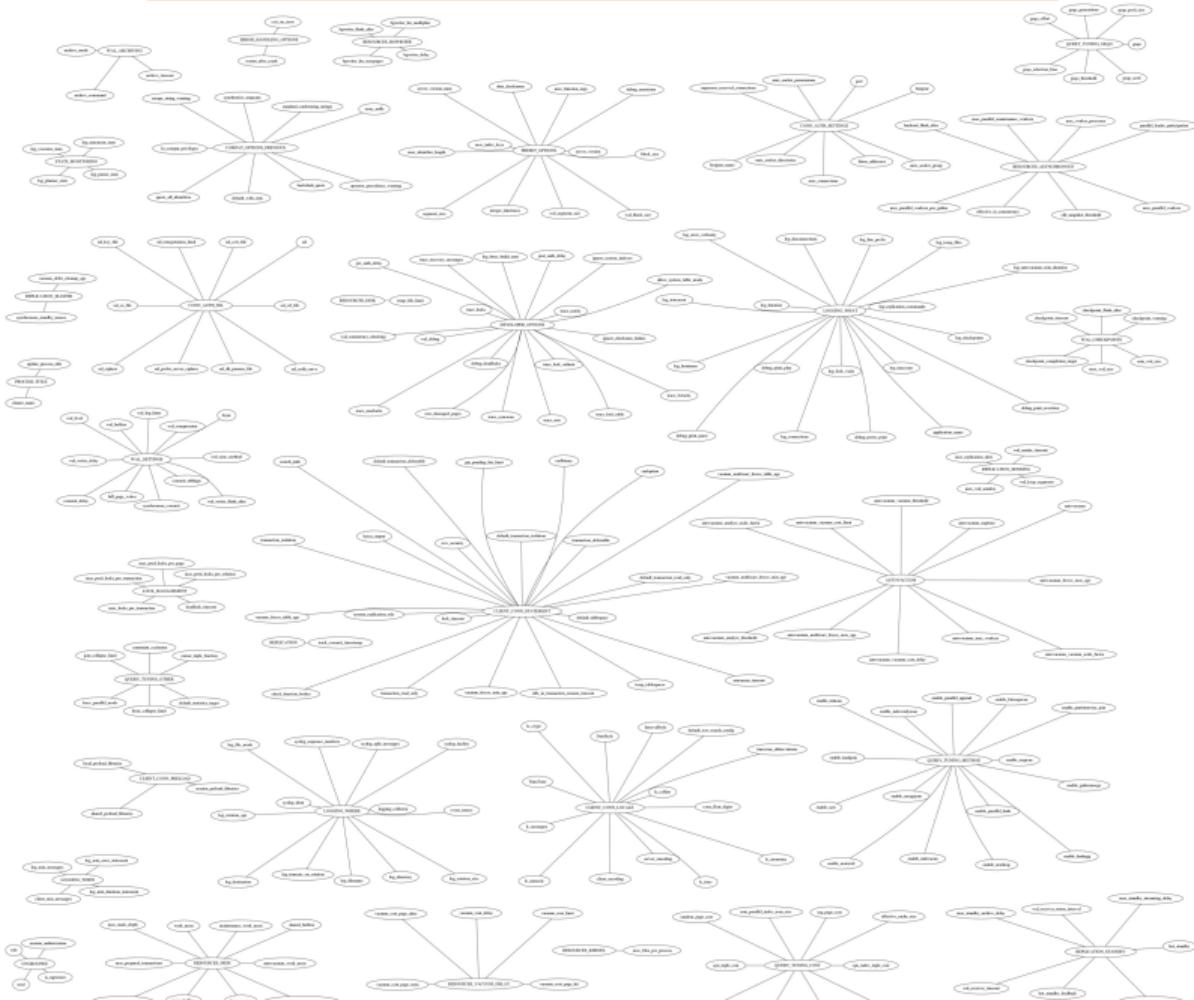
---

FRIENDSHIP

DMITRY DOLGOV

20-03-2018







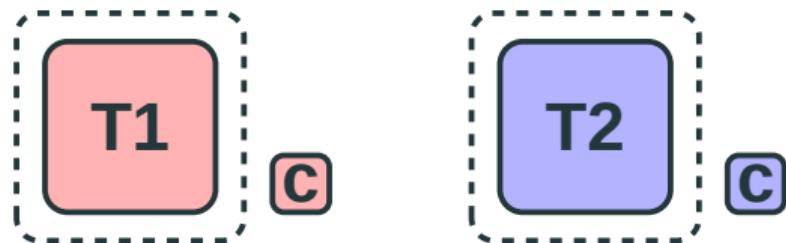
# Execution scheduling

```
# Experiment 1
transaction type: pg_long.sql
latency average = 1312.903 ms

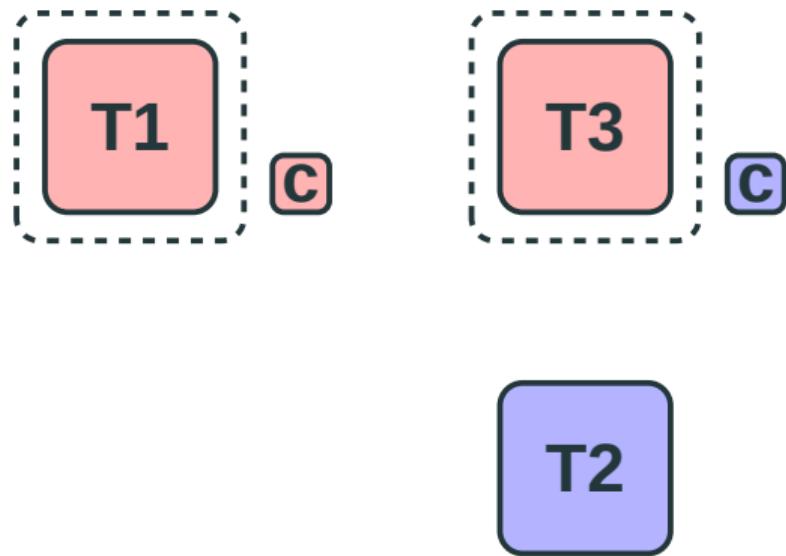
# Experiment 2
SQL script 1: pg_long.sql
- weight: 1 (targets 50.0% of total)
- latency average = 1426.928 ms

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

# Scheduling



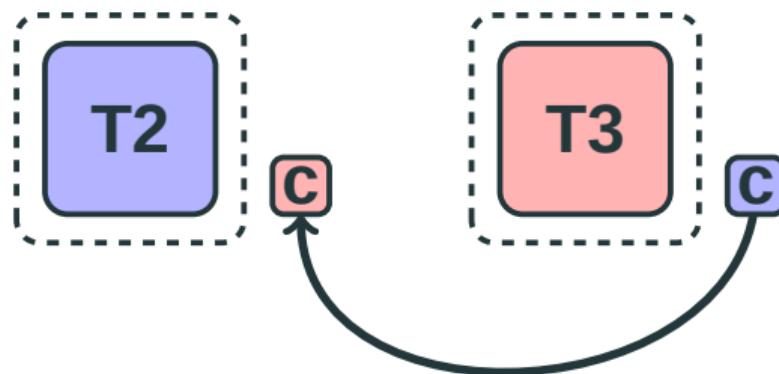
# Scheduling



# Scheduling



# Scheduling



```
# 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 [tid/pid]	wait time (msec)	sch delay (msec)	run time (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	

# Tunables

# Tunables

→ /proc/sys/kernel/sched\_migration\_cost\_ns

# Tunables

→ /proc/sys/kernel/sched\_migration\_cost\_ns  
→ 500000

# Tunables

- /proc/sys/kernel/sched\_migration\_cost\_ns
- 500000
- /proc/sys/kernel/sched\_wakeup\_granularity\_ns

# Tunables

- /proc/sys/kernel/sched\_migration\_cost\_ns
- 500000
- /proc/sys/kernel/sched\_wakeup\_granularity\_ns
- 3000000

# Tunables

- /proc/sys/kernel/sched\_migration\_cost\_ns
- 500000
- /proc/sys/kernel/sched\_wakeup\_granularity\_ns
- 3000000
- /proc/sys/kernel/sched\_min\_granularity\_ns

# Tunables

- /proc/sys/kernel/sched\_migration\_cost\_ns
- 500000
- /proc/sys/kernel/sched\_wakeup\_granularity\_ns
- 3000000
- /proc/sys/kernel/sched\_min\_granularity\_ns
- 2250000

# Tunables

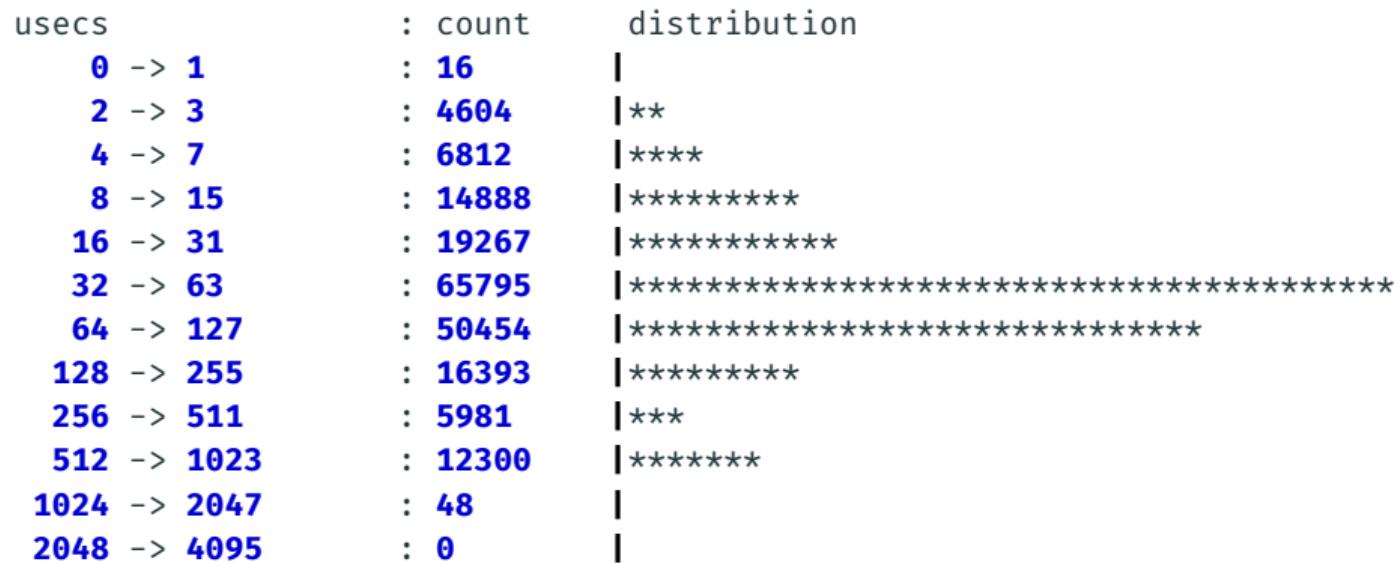
- /proc/sys/kernel/sched\_migration\_cost\_ns
- 500000
- /proc/sys/kernel/sched\_wakeup\_granularity\_ns
- 3000000
- /proc/sys/kernel/sched\_min\_granularity\_ns
- 2250000
- /proc/sys/kernel/sched\_latency\_ns

# Tunables

- /proc/sys/kernel/sched\_migration\_cost\_ns
- 500000
- /proc/sys/kernel/sched\_wakeup\_granularity\_ns
- 3000000
- /proc/sys/kernel/sched\_min\_granularity\_ns
- 2250000
- /proc/sys/kernel/sched\_latency\_ns
- 18000000

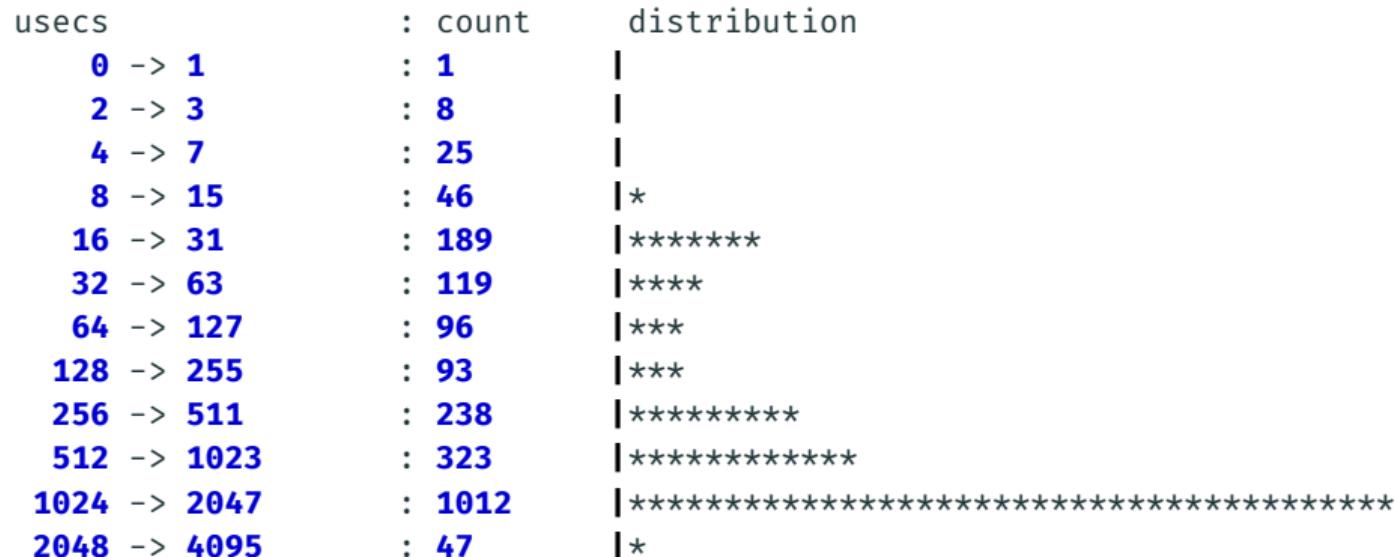
# pgbench and pg\_dump

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

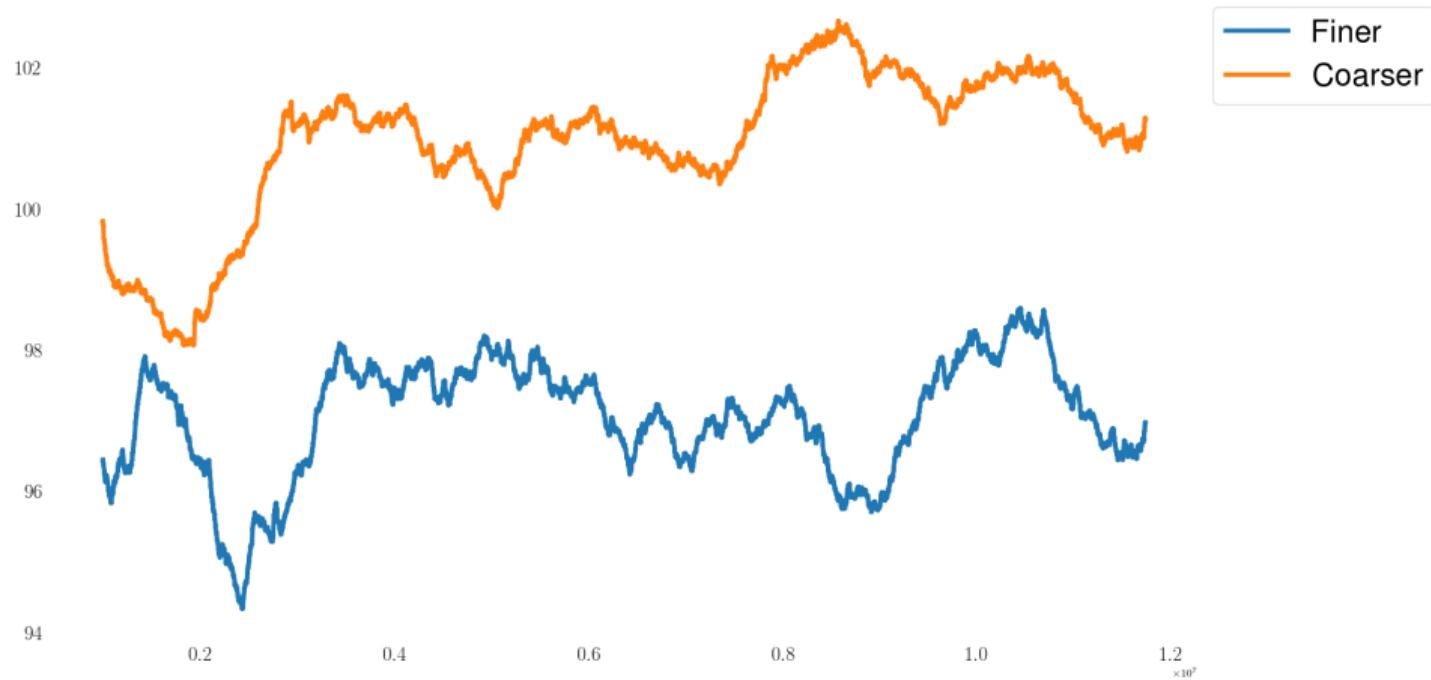


# pgbench and pg\_dump

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



# Wakeup granularity, microsec



# HyperThreading

Intel® 64 and IA-32 Architectures Optimization Reference Manual

# HyperThreading

→ Less deviation for latency

Intel® 64 and IA-32 Architectures Optimization Reference Manual

# HyperThreading

- Less deviation for latency
- Share execution state and cache

Intel® 64 and IA-32 Architectures Optimization Reference Manual

# HyperThreading

- Less deviation for latency
- Share execution state and cache
- Spin locks have significant impact

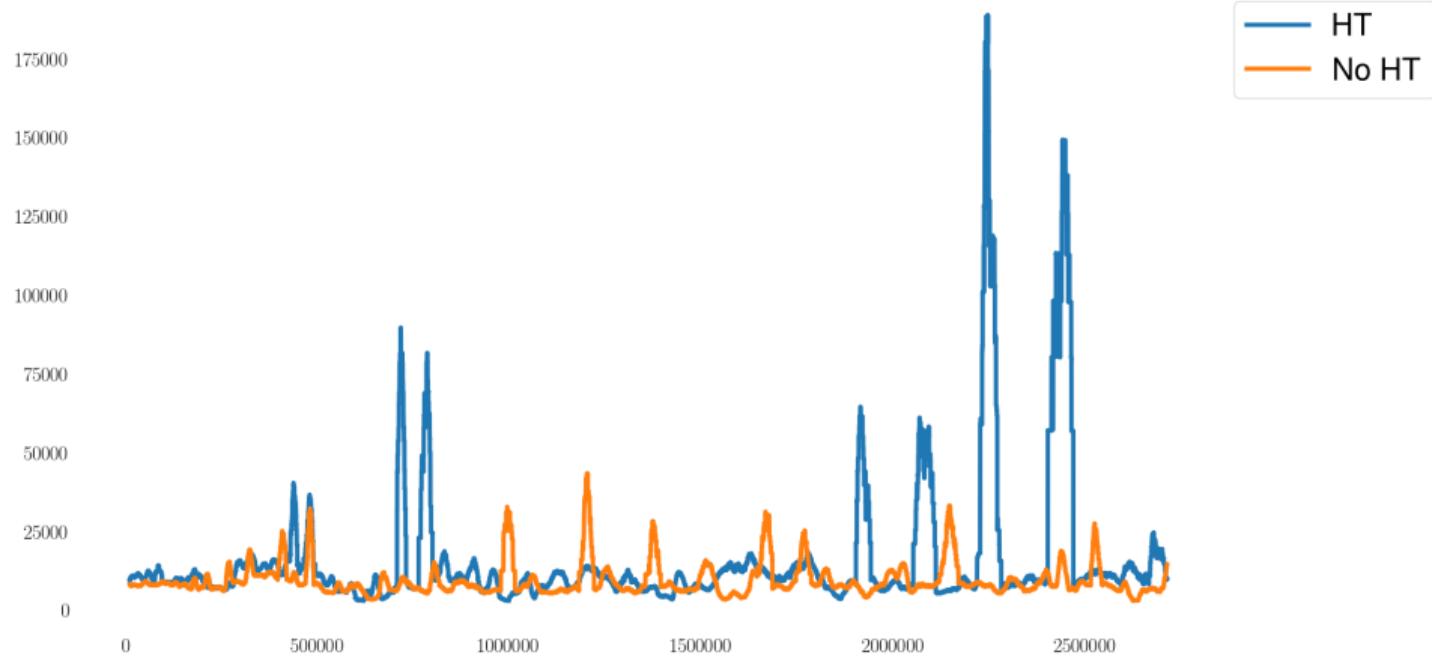
Intel® 64 and IA-32 Architectures Optimization Reference Manual

# HyperThreading

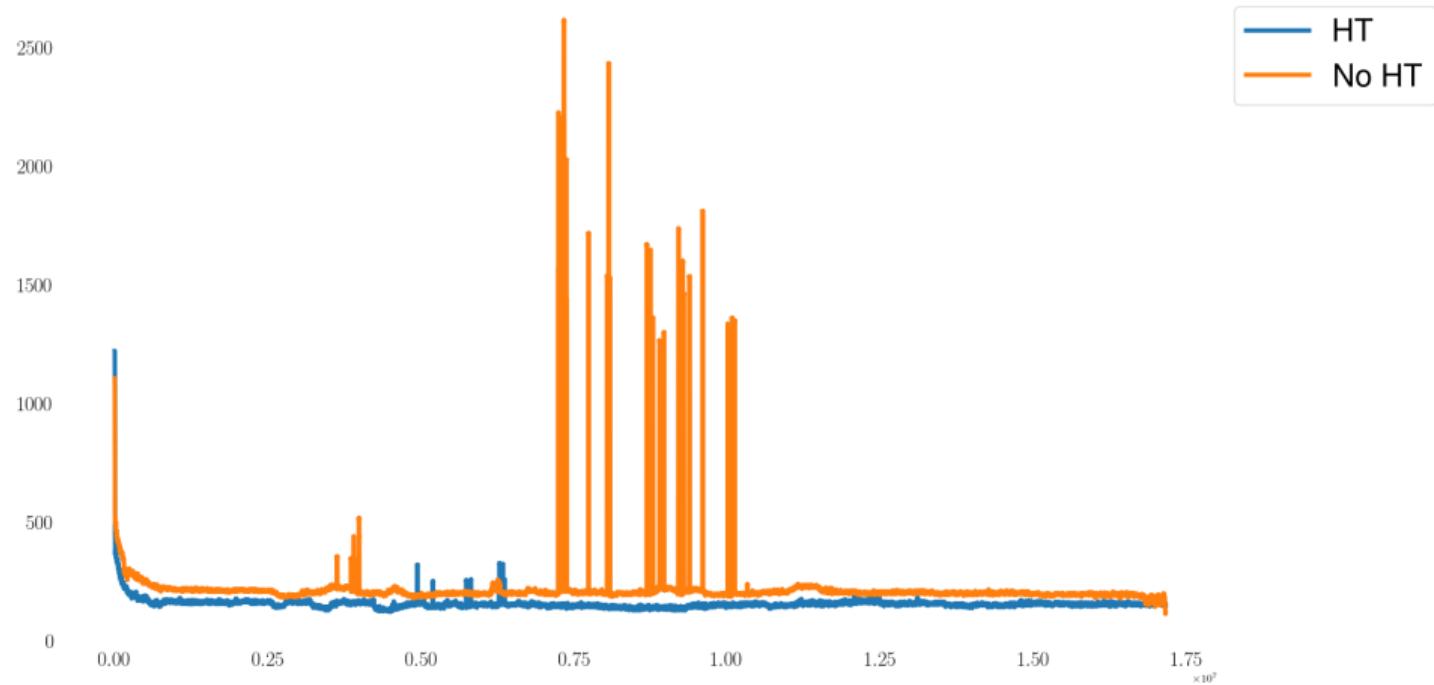
- Less deviation for latency
- Share execution state and cache
- Spin locks have significant impact
- PAUSE instruction (skylake latency 140 cycles)

Intel® 64 and IA-32 Architectures Optimization Reference Manual

# Latency rolling standard deviation, r/w



# Latency rolling standard deviation, readonly



# Virtualization?

# Timekeeping

Timekeeping in VMware Virtual: Information Guide

# Timekeeping

→ Statistical sampling  
(occasional incorrect charging)

Timekeeping in VMware Virtual: Information Guide

# Timekeeping

- Statistical sampling  
(occasional incorrect charging)
- Exact measurement (TSC time drift)

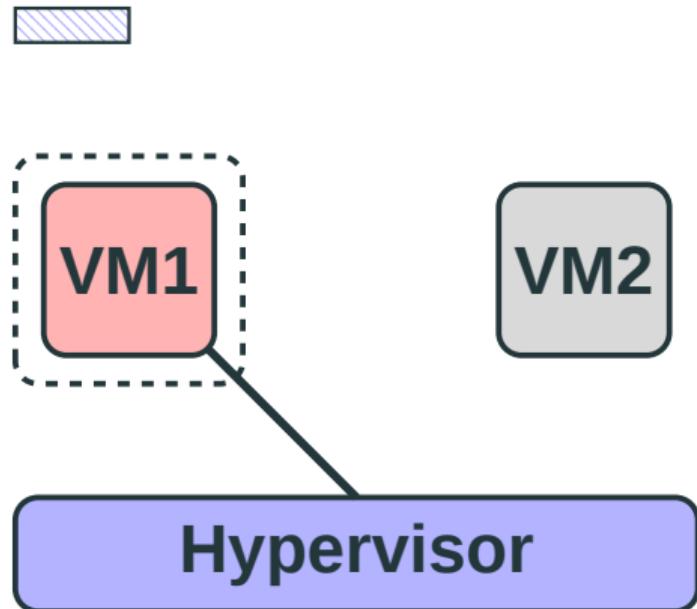
Timekeeping in VMware Virtual: Information Guide

# Timekeeping

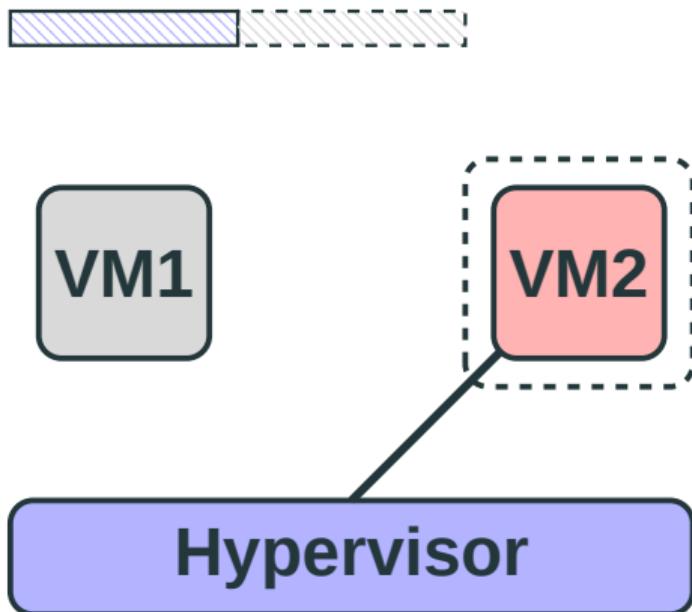
- Statistical sampling  
(occasional incorrect charging)
- Exact measurement (TSC time drift)
- `/sys/devices/system/clocksource/clocksource0/`

Timekeeping in VMware Virtual: Information Guide

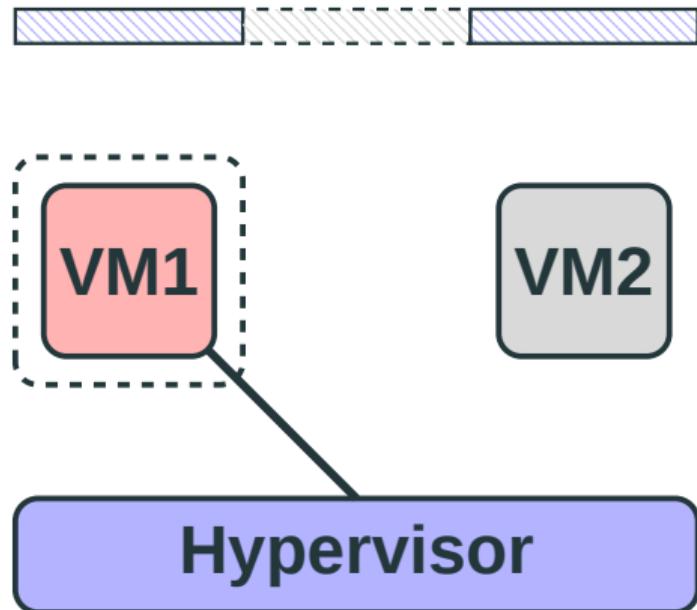
# Scheduling



# Scheduling



# Scheduling

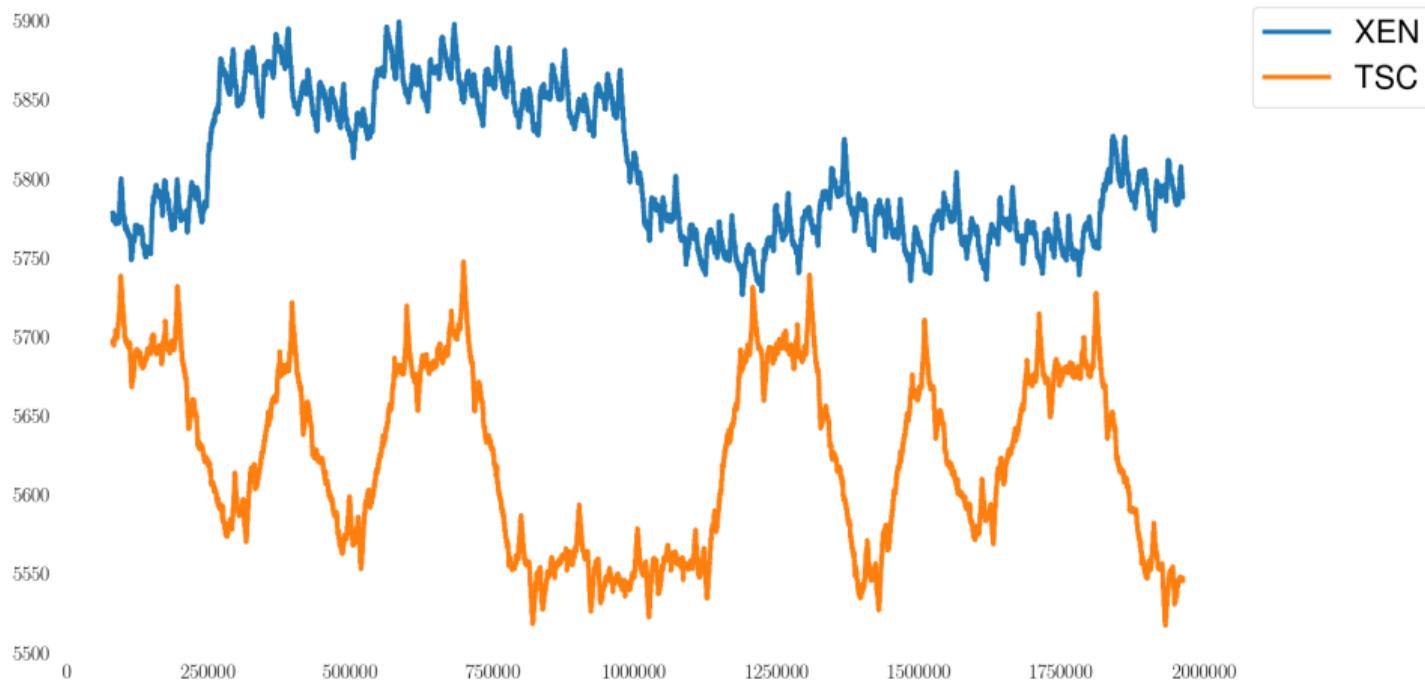


## vDSO

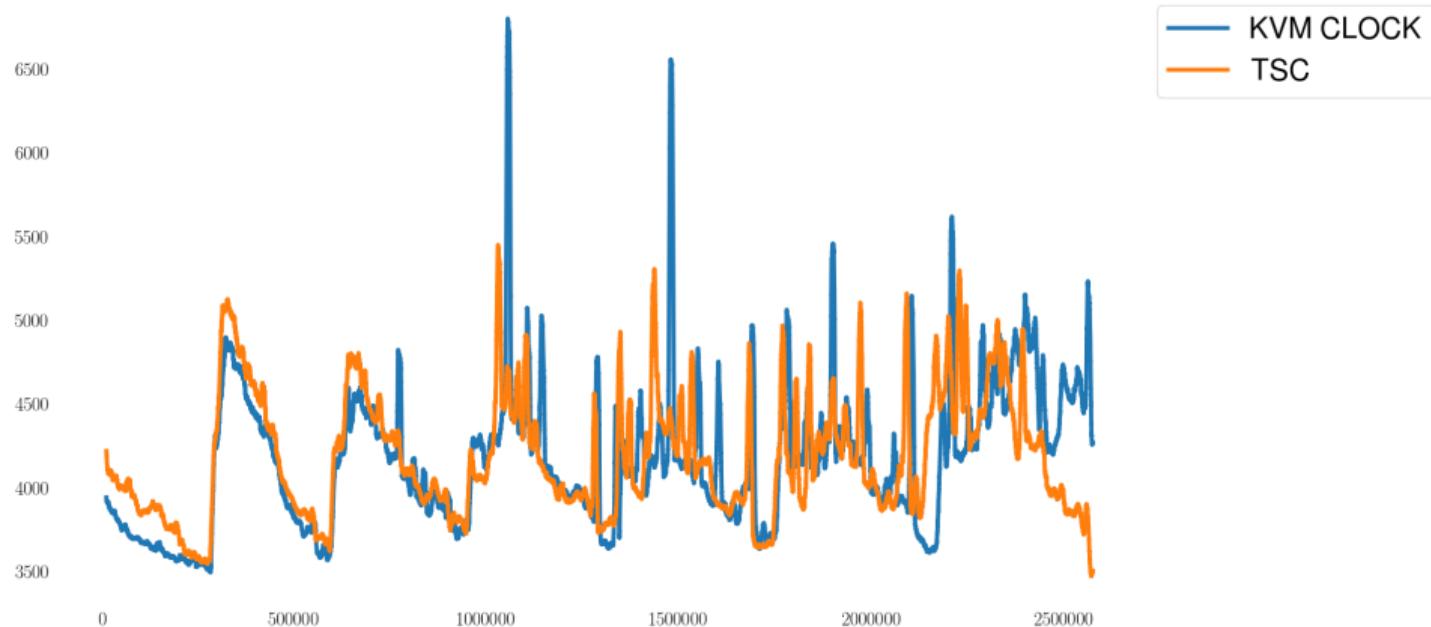
- gettimeofday
- clock\_gettime
- XEN doesn't support vDSO for them
- unnecessary context switches to a kernel

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

# Latency m4.xlarge XEN/TSC, r/w

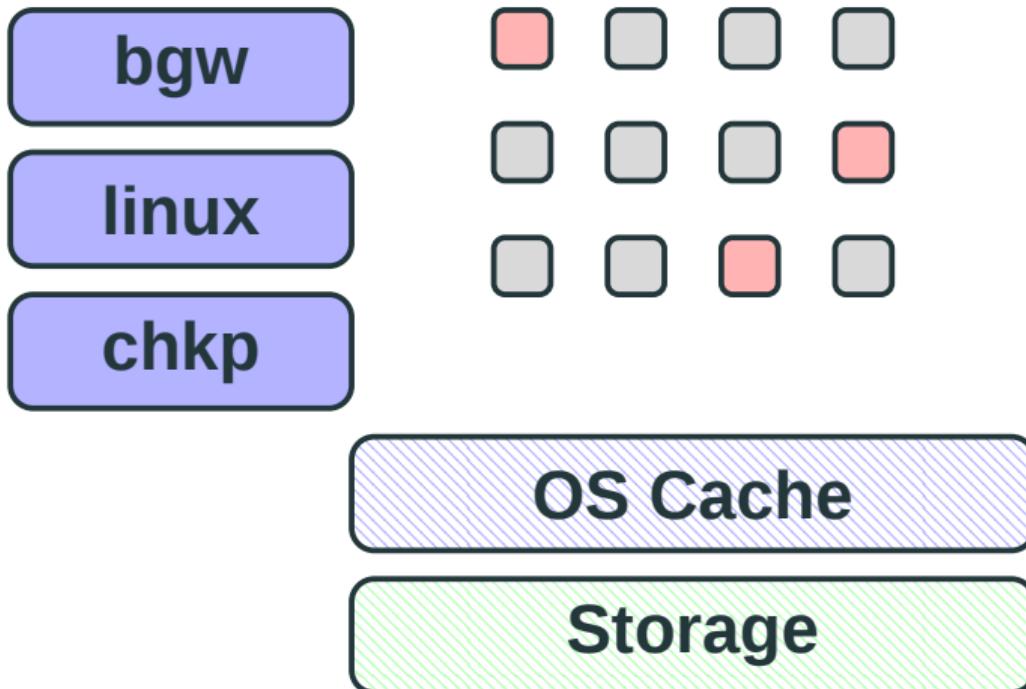


# Latency m5.xlarge KVM/TSC, r/w

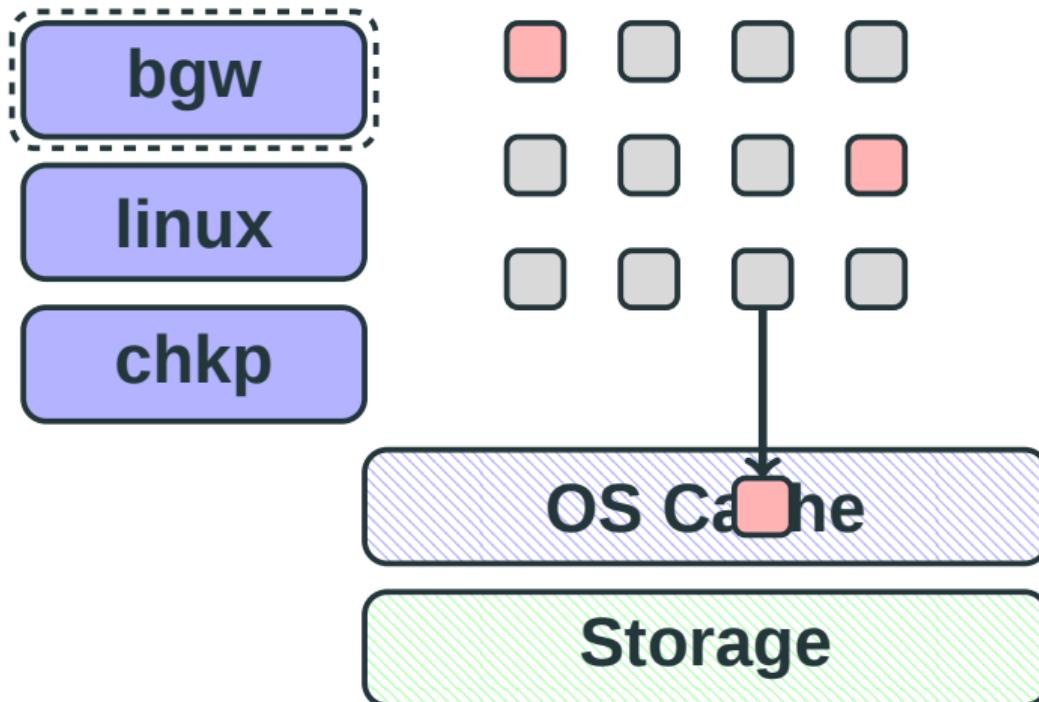


# Memory

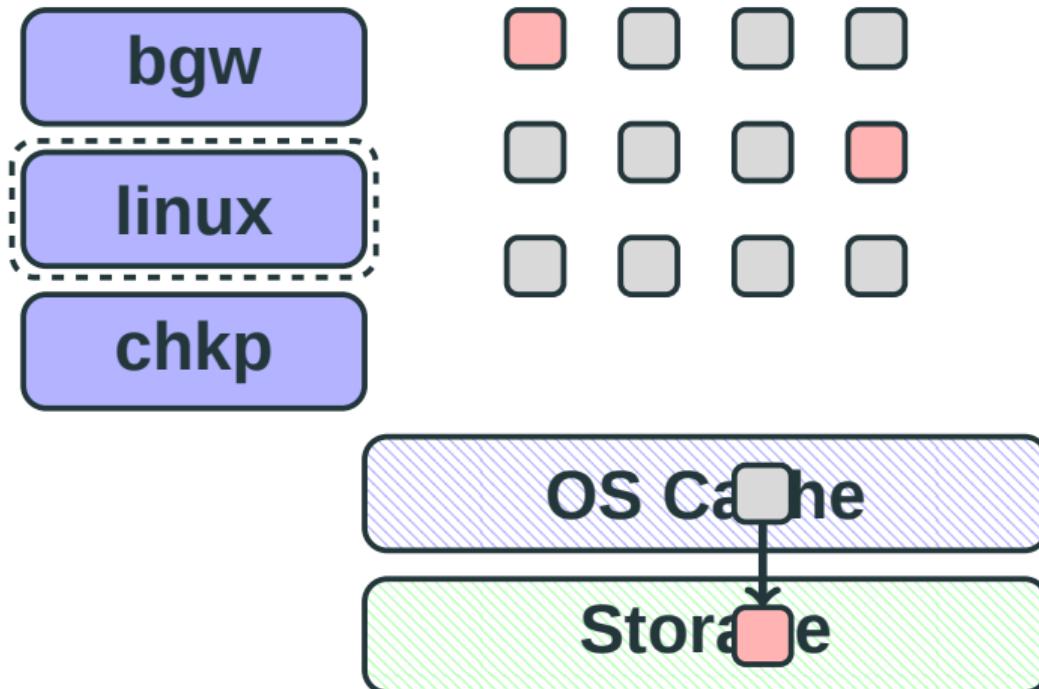
# Dirty pages



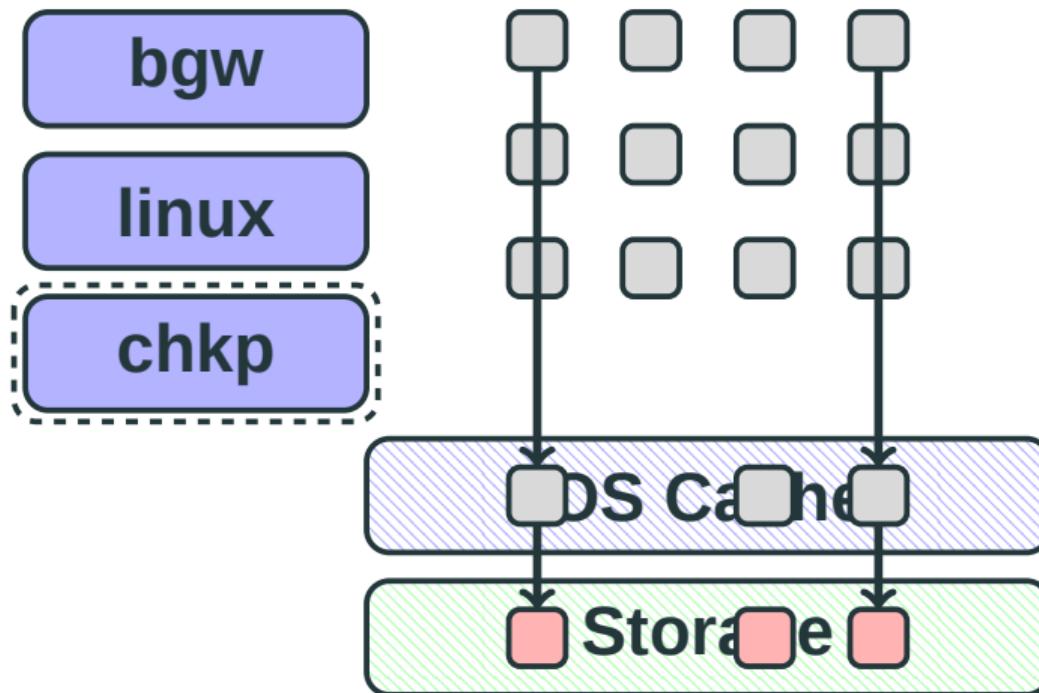
# Dirty pages



# Dirty pages



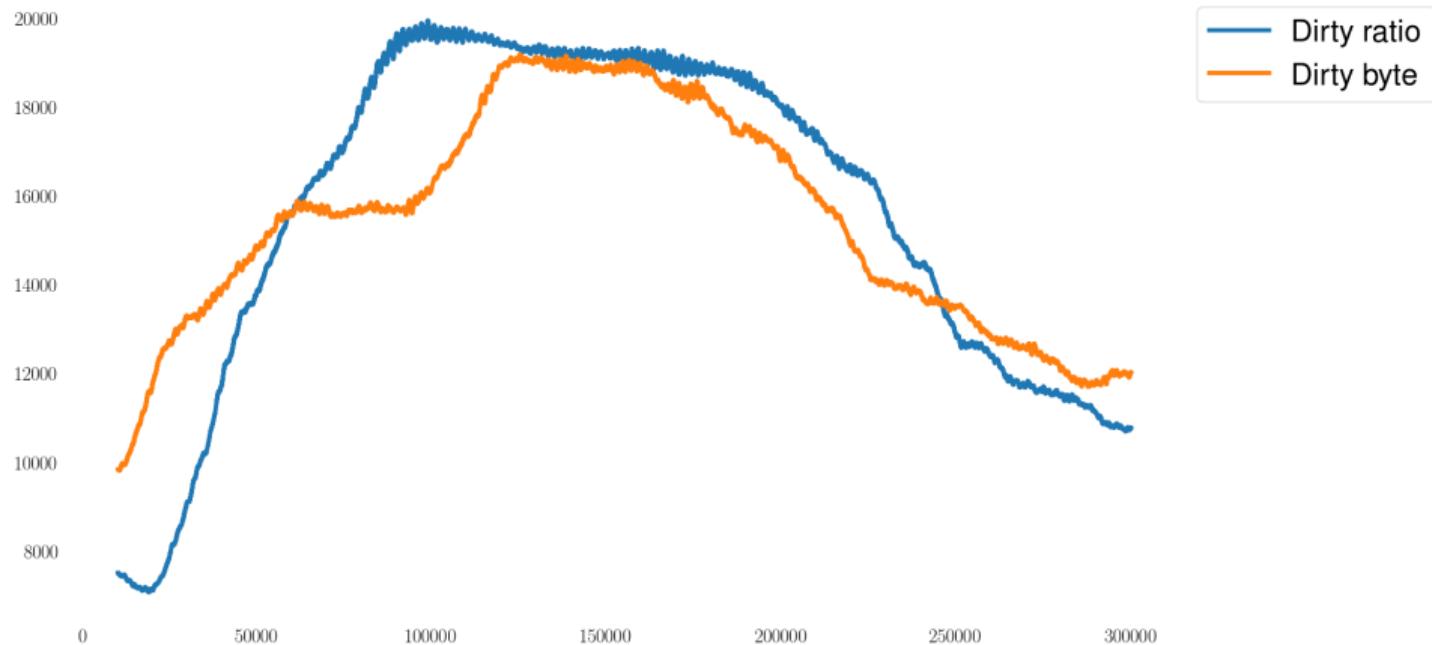
# Dirty pages



## Dirty pages, r/w

- vm.dirty\_ratio
- vm.dirty\_background\_ratio
- vm.dirty\_bytes
- vm.dirty\_background\_bytes

# Dirty pages



# Containers?

# cgroups controllers

- cpu,cpuacct
- cpuset
- memory
- devices
- freezer
- net\_cls
- rdma
- blkio
- perf\_event
- net\_prio
- hugetlb
- pids
- rdma

## **CONFIG\_MEMCG\_KMEM**

- Enabled in modern versions
- PostgreSQL requires contiguous memory for shared buffers
- It's being allocated using slab
- `memory.kmem.limit_in_bytes` is too high

# Host, normal

Zone: Normal

Free KiB in zone: **807232.00**

Fragment size	Free fragments
<b>4096</b>	<b>29612</b>
<b>8192</b>	<b>23308</b>
<b>16384</b>	<b>13495</b>

# Host with a container

Zone: Normal

Free KiB in zone: **109700.00**

Fragment size	Free fragments
<b>4096</b>	<b>3405</b>
<b>8192</b>	<b>7082</b>
<b>16384</b>	<b>1954</b>

## Bad neighbor

- buddy allocator can fail  
to find a page of proper size
- kernel will start a compaction process
- compaction implementation  
knows nothing about cgroups

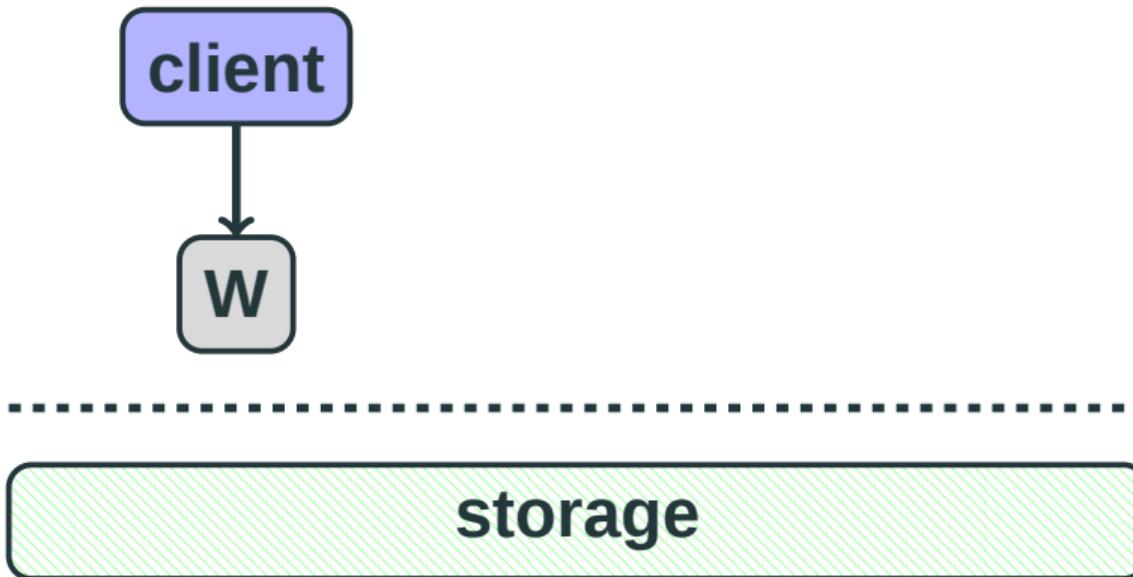
# Storage IO

# WAL

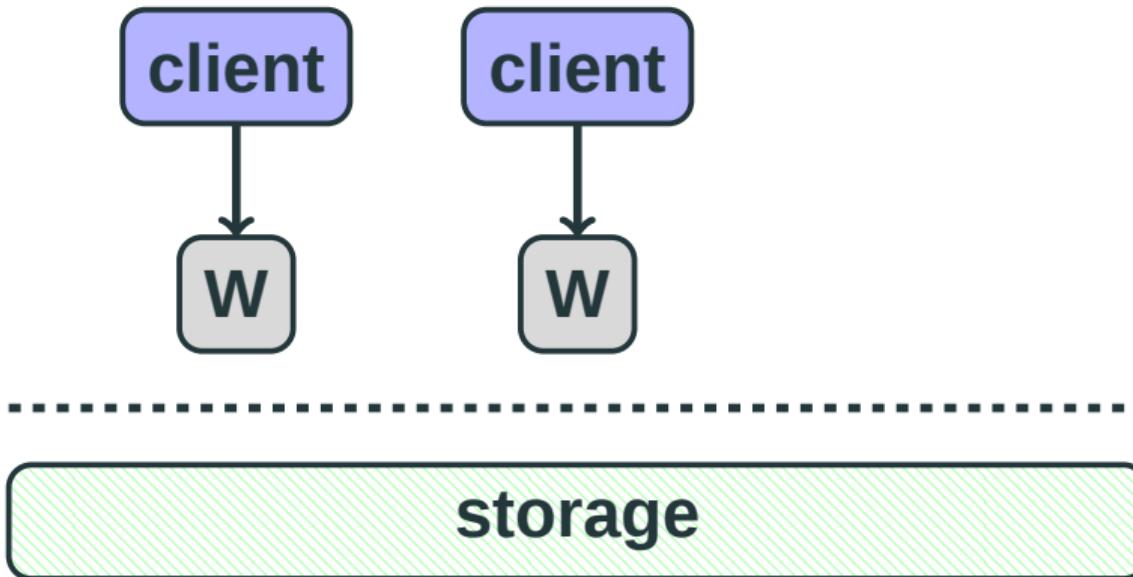
client

storage

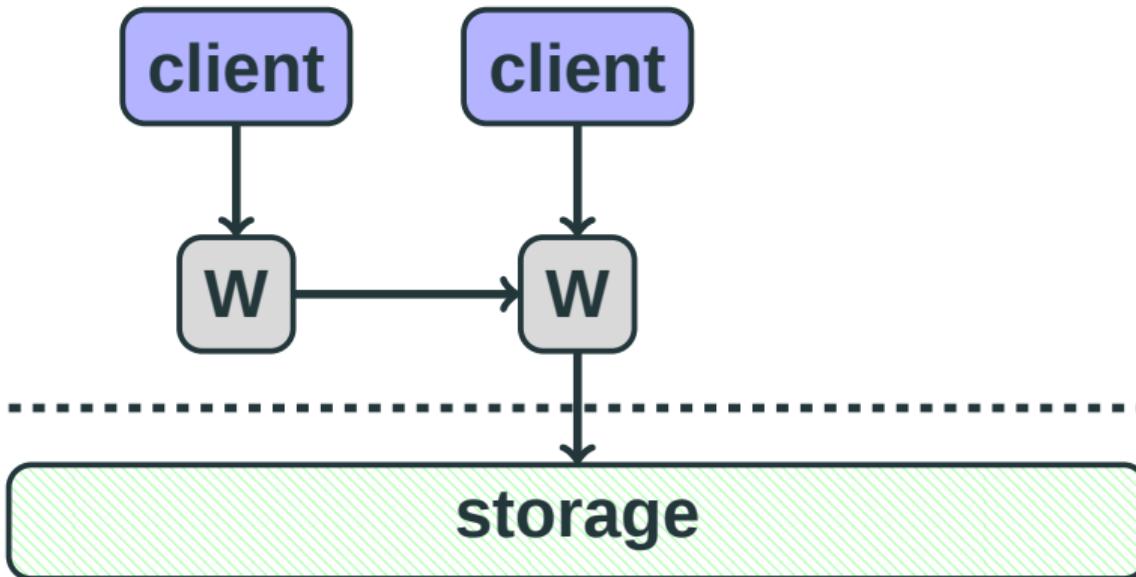
# WAL



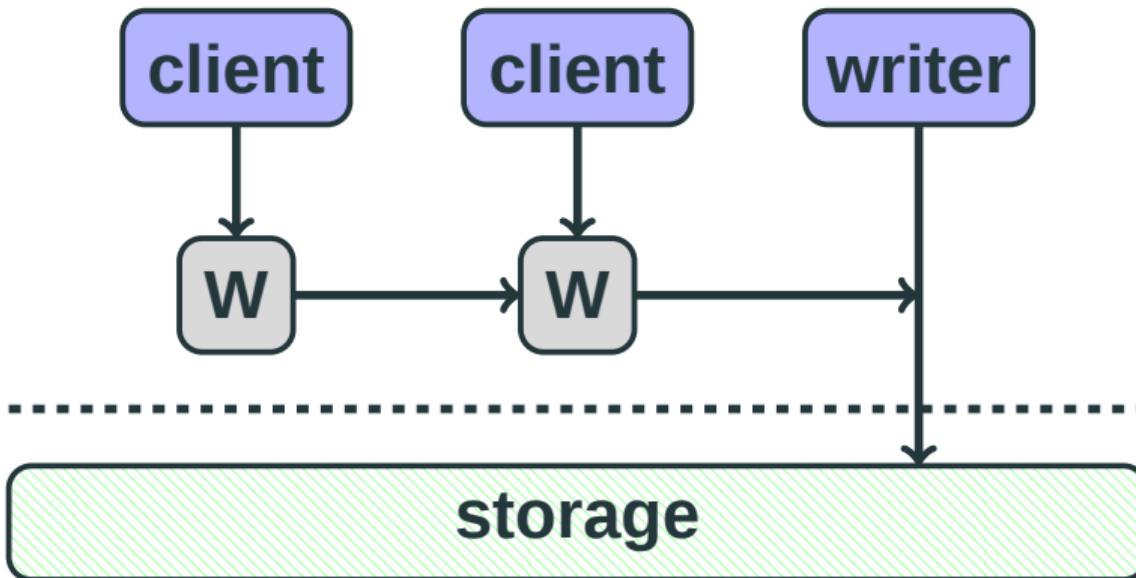
# WAL



# WAL



# WAL



# WAL

- bgwriter write to WAL  
(transaction snapshots for replication)
- with logical decoding turned on  
more data is going to WAL

## NVMe

- better for resource sharing (PCI express) under the virtualization
- /sys/block/sda/queue/scheduler [noop|none]
- DSM operations

## NVMe DSM

- Expected lifetime
- Sequential write/read prepare
- Access frequency

## DSM support

- Command DWORD 11 in ioctl
- fcntl SET\_FILE\_RW\_HINT
- nvme-cli
- Specify a start block and a range length

NVM Express Revision 1.3c May 24, 2018

```
# get a start block
hdparm --fibmap data_file
data_file:
filesystem blocksize 4096, begins at LBA 0;
assuming 512 byte sectors.
byte_offset begin_LBA end_LBA sectors
      0    55041560   55041567        8

# set dsm for sequential read optimized
nvme dsm /dev/nvme1n01 --idr --slbs=55041560 --blocks=8
```

# Containers?

## **blkio controller**

- CFQ & throttling policy (generic block layer)
- No weight related options will work without CFQ
- Advisable io scheduler for SSD is noop/none
- Block layer do sampling to enforce throttling

8388 8388 postgres blk\_throtl\_bio  
blk\_throtl\_bio+0x1 [kernel]  
dm\_make\_request+0x80 [kernel]  
generic\_make\_request+0xf6 [kernel]  
submit\_bio+0x7d [kernel]  
blkdev\_issue\_flush+0x68 [kernel]  
ext4\_sync\_file+0x310 [kernel]  
vfs\_fsync\_range+0x4b [kernel]  
do\_fsync+0x3d [kernel]  
sys\_fdatasync+0x13 [kernel]  
fdatasync+0x10 [libc-2.24.so]  
XLogBackgroundFlush+0x17e [postgres]  
WalWriterMain+0x1cb [postgres]  
PostmasterMain+0xfea [postgres]

## **throttle\_sample\_time**

This is the time window that blk-throttle samples data, in millisecond. blk-throttle makes decision based on the samplings. Lower time means cgroups have more smooth throughput, but higher CPU overhead. This exists only when CONFIG\_BLK\_DEV\_THROTTLING\_LOW is enabled.

## blkio

On traditional cgroup hierarchies, relationships between different controllers cannot be established making it impossible for writeback to operate accounting for cgroup resource restrictions and all writeback IOs are attributed to the root cgroup.

<https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git>

## Questions?

⌚ [github.com/erthalion](https://github.com/erthalion)

🐦 [@erthalion](https://twitter.com/erthalion)

✉️ [dmitrii.dolgov at zalando dot de](mailto:dmitrii.dolgov@zalando.de)

✉️ [9erthalion6 at gmail dot com](mailto:9erthalion6@gmail.com)