Where did my memory go?

Frits Hoogland
Yugabyte

PostgreSQL relies on the OS for memory

In most cases 'the OS' will be Linux.

Linux memory usage can be seen in many ways.

A common way is: /proc/meminfo

```
[vagrant@pg15 ~]$ cat /proc/meminfo
MemTotal:        1856020 kB
MemFree:         1471340 kB
MemAvailable:    1555788 kB
Buffers:            4204 kB
Cached:           203252 kB
SwapCached:            0 kB
Active:            60820 kB
Inactive:         215696 kB
Active(anon):        784 kB
Inactive(anon):    76888 kB
Active(file):      60036 kB
Inactive(file):   138808 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:       2154492 kB
SwapFree:        2154492 kB
Dirty:                32 kB
Writeback:             0 kB
AnonPages:         69064 kB
Mapped:            93892 kB
Shmem:              8612 kB
KReclaimable:      33040 kB
Slab:              68128 kB
SReclaimable:      33040 kB
SUnreclaim:        35088 kB
KernelStack:        2224 kB
PageTables:         5856 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     3082500 kB
Committed_AS:     324464 kB
VmallocTotal:   34359738367 kB
VmallocUsed:           0 kB
VmallocChunk:          0 kB
Percpu:              992 kB
HardwareCorrupted:     0 kB
AnonHugePages:     14336 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
FileHugePages:         0 kB
FilePmdMapped:         0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:               0 kB
DirectMap4k:       90048 kB
DirectMap2M:     2007040 kB
```

PostgreSQL relies on the OS for memory

Problem: lots of entries.

/proc/meminfo is a motley gathering of memory related statistics.

Multiple views: (in)active, file/anon, memory/swap, hugepages (pages!), virtual, etc.

```
[vagrant@pg15 ~]$ cat /proc/meminfo
MemTotal:        1856020 kB
MemFree:         1471340 kB
MemAvailable:    1555788 kB
Buffers:            4204 kB
Cached:           203252 kB
SwapCached:            0 kB
Active:            60820 kB
Inactive:         215696 kB
Active(anon):        784 kB
Inactive(anon):    76888 kB
Active(file):      60036 kB
Inactive(file):   138808 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:       2154492 kB
SwapFree:        2154492 kB
Dirty:                32 kB
Writeback:             0 kB
AnonPages:         69064 kB
Mapped:            93892 kB
Shmem:              8612 kB
KReclaimable:      33040 kB
Slab:              68128 kB
SReclaimable:      33040 kB
SUnreclaim:        35088 kB
KernelStack:        2224 kB
PageTables:         5856 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     3082500 kB
Committed_AS:     324464 kB
VmallocTotal:   34359738367 kB
VmallocUsed:           0 kB
VmallocChunk:          0 kB
Percpu:              992 kB
HardwareCorrupted:     0 kB
AnonHugePages:     14336 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
FileHugePages:         0 kB
FilePmdMapped:         0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:               0 kB
DirectMap4k:       90048 kB
DirectMap2M:     2007040 kB
```

Linux allows memory oversubscription

Setting: `vm.overcommit_memory`

Default: 0 (heuristic overcommit) -> OOM killer.

Best: 2 (don't overcommit) -> Error on memory allocation (single backend).

Test!

Linux - meminfo - Memory hierarchy v1.0

Virtual Memory (not a statistic)

- MemTotal
  - MemFree
  - Slab
  - PageTables
  - KernelStack
  - SwapCached
  - AnonPages
  - Cached
    - Mapped
      - Shmem (SysV /dev/zero mmapped)
    - Dirty
- SwapTotal
  - SwapFree

Linux - meminfo - Memory hierarchy v1.0

Virtual Memory (not a statistic)

MemTotal

MemFree

Slab

PageTables

KernelStack

SwapCached

AnonPages

Cached

Mapped

Shmem (SysV /dev/zero mmapped)

Dirty

SwapTotal

SwapFree

Linux - meminfo - Memory hierarchy v1.0

Virtual Memory (not a statistic)

MemTotal

MemFree

Slab

PageTables

KernelStack

SwapCached

AnonPages

Cached

Mapped

Shmem (SysV /dev/zero mmapped)

Dirty

SwapTotal

SwapFree

Linux - meminfo - Memory hierarchy v1.0

Virtual Memory (not a statistic)

MemTotal

MemFree

Slab

PageTables

KernelStack

SwapCached

AnonPages

Cached

Mapped

Shmem (SysV /dev/zero mmapped)

Dirty

SwapTotal

SwapFree

Linux - meminfo - Memory hierarchy v1.0

Linux - meminfo - Memory hierarchy v1.0

Virtual Memory (not a statistic)

MemTotal

MemFree

Slab

PageTables

KernelStack

SwapCached

AnonPages

Cached

Mapped

Shmem (SysV /dev/zero mmapped -- ??)

Dirty

SwapTotal

SwapFree

Linux - meminfo - Memory hierarchy v1.0

Virtual Memory (not a statistic)

MemTotal

MemFree

Slab

PageTables

KernelStack

SwapCached

AnonPages

Cached

Mapped

Shmem (SysV /dev/zero mmapped)

Dirty

SwapTotal

SwapFree

Linux - meminfo - Memory hierarchy v1.0

Virtual Memory (not a statistic)

MemTotal

MemFree

Slab

PageTables

KernelStack

SwapCached

AnonPages

Cached

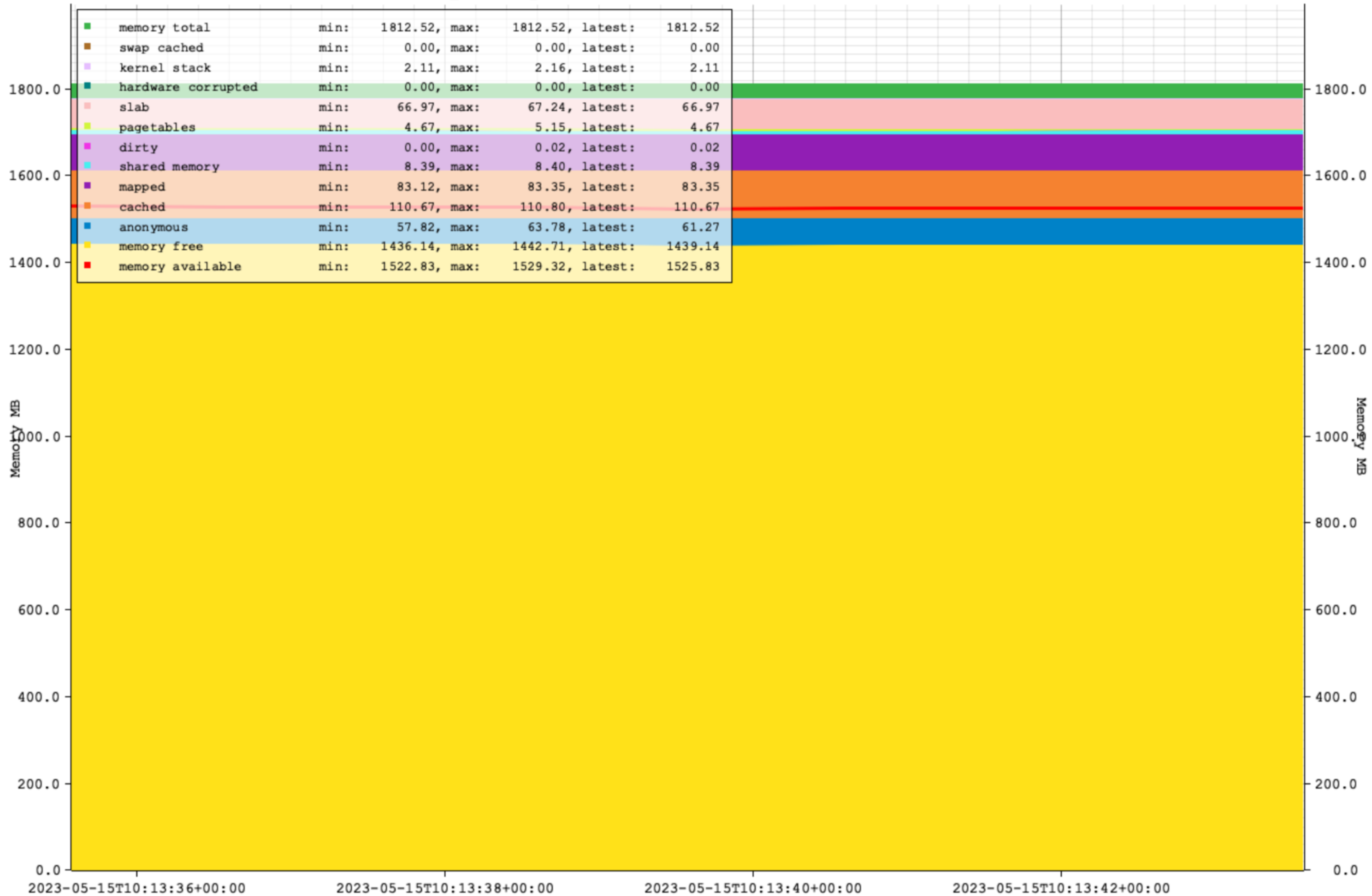
Mapped

Shmem (SysV /dev/zero mmapped)

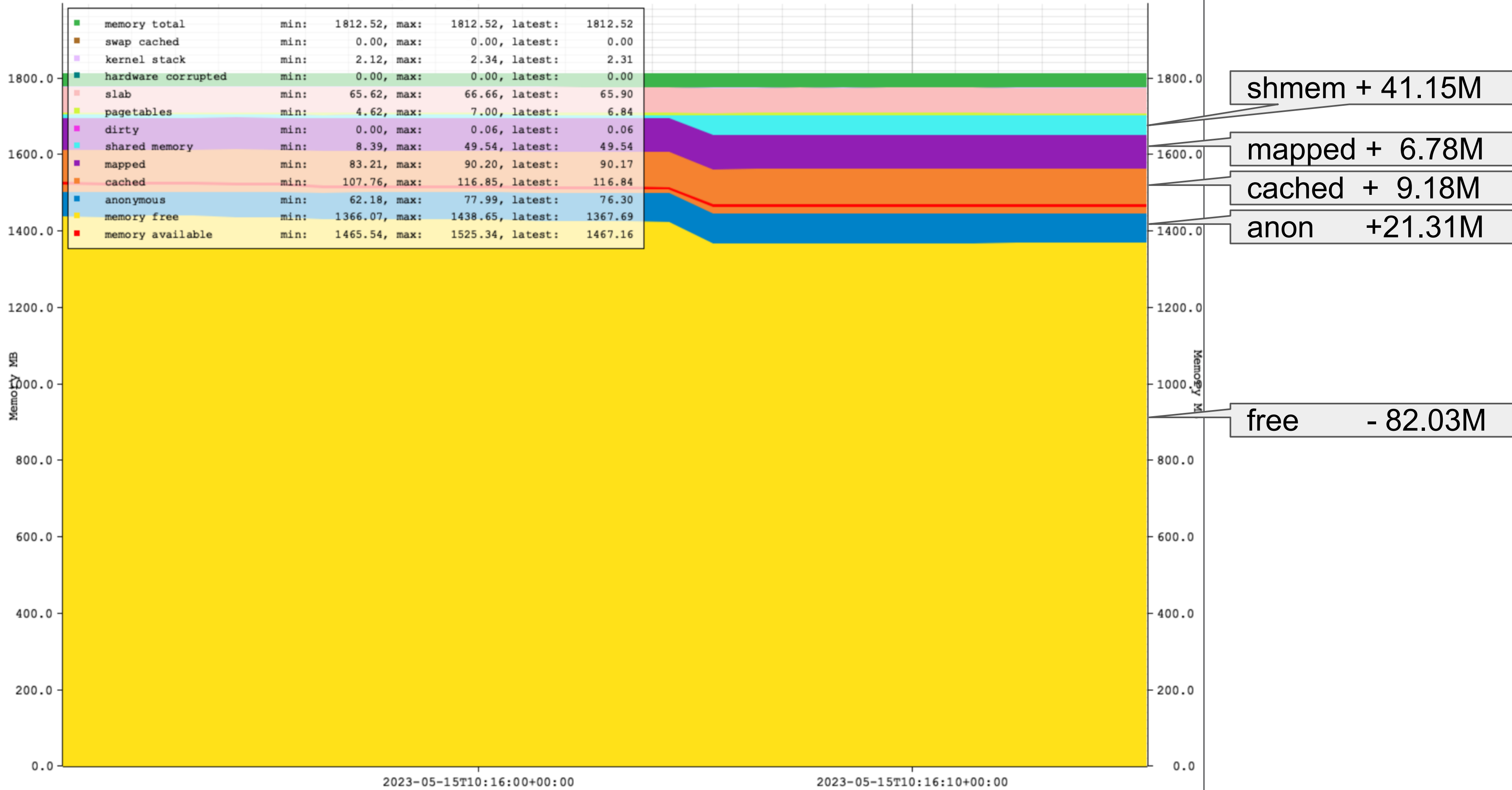
Dirty

SwapTotal

SwapFree



Memory usage: 192.168.66.100:9300:metrics
memory total min: 1812.52, max: 1812.52, latest: 1812.52
swap cached min: 0.00, max: 0.00, latest: 0.00
kernel stack min: 2.11, max: 2.16, latest: 2.11
hardware corrupted min: 0.00, max: 0.00, latest: 0.00
slab min: 66.97, max: 67.24, latest: 66.97
pagetables min: 4.67, max: 5.15, latest: 4.67
dirty min: 0.00, max: 0.02, latest: 0.02
shared memory min: 8.39, max: 8.40, latest: 8.39
mapped min: 83.12, max: 83.35, latest: 83.35
cached min: 110.67, max: 110.80, latest: 110.67
anonymous min: 57.82, max: 63.78, latest: 61.27
memory free min: 1436.14, max: 1442.71, latest: 1439.14
memory available min: 1522.83, max: 1529.32, latest: 1525.83
Memory MB
2023-05-15T10:13:36+00:00
2023-05-15T10:13:38+00:00
2023-05-15T10:13:40+00:00
2023-05-15T10:13:42+00:00



Memory usage: 192.168.66.100:9300:metrics
memory total min: 1812.52, max: 1812.52, latest: 1812.52
swap cached min: 0.00, max: 0.00, latest: 0.00
kernel stack min: 2.12, max: 2.34, latest: 2.31
hardware corrupted min: 0.00, max: 0.00, latest: 0.00
slab min: 65.62, max: 66.66, latest: 65.90
pagetables min: 4.62, max: 7.00, latest: 6.84
dirty min: 0.00, max: 0.06, latest: 0.06
shared memory min: 8.39, max: 49.54, latest: 49.54
mapped min: 83.21, max: 90.20, latest: 90.17
cached min: 107.76, max: 116.85, latest: 116.84
anonymous min: 62.18, max: 77.99, latest: 76.30
memory free min: 1366.07, max: 1438.65, latest: 1367.69
memory available min: 1465.54, max: 1525.34, latest: 1467.16
Memory MB
2023-05-15T10:16:00+00:00
2023-05-15T10:16:10+00:00
shmem + 41.15M
mapped + 6.78M
cached + 9.18M
anon +21.31M
free - 82.03M

The shared memory segment from 'smaps':

```
7f5187787000-7f51ca8d3000 rw-s 00000000 00:01 33813                      /dev/zero (deleted)
Size:            1099056 kB
KernelPageSize:        4 kB
MMUPageSize:           4 kB
Rss:               41012 kB
Pss:               26659 kB
Shared_Clean:          0 kB
Shared_Dirty:      25540 kB
Private_Clean:         0 kB
Private_Dirty:     15472 kB
Referenced:        41012 kB
Anonymous:             0 kB
LazyFree:              0 kB
AnonHugePages:         0 kB
ShmemPmdMapped:        0 kB
FilePmdMapped:         0 kB
Shared_Hugetlb:        0 kB
Private_Hugetlb:       0 kB
Swap:                  0 kB
SwapPss:               0 kB
Locked:                0 kB
THPeligible:    0
VmFlags: rd wr sh mr mw me ms sd
```

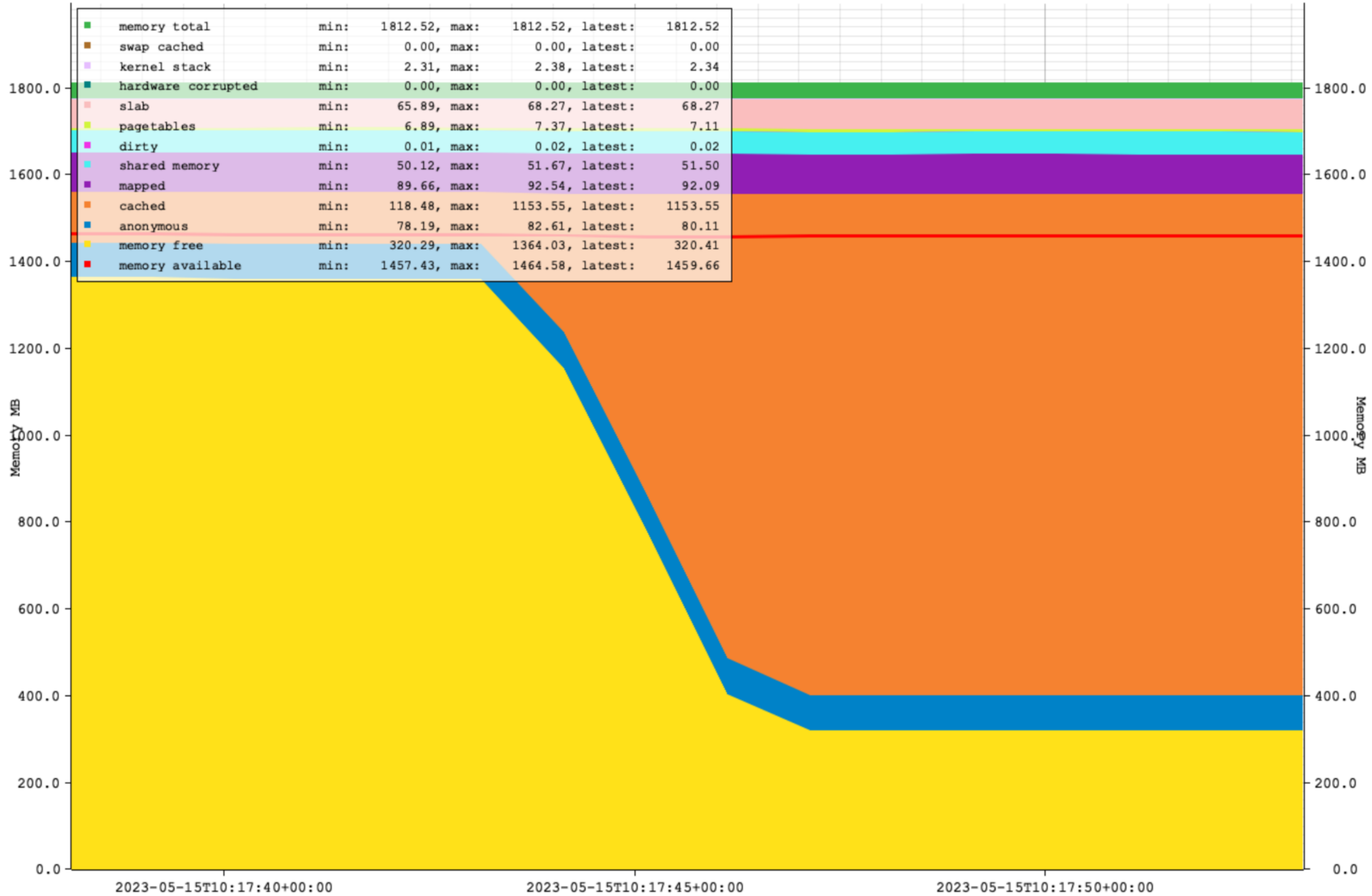
Let's test!

```
-- precreated table:
create table t( id int primary key, f1 text);
insert into t select id, repeat('x',16384) from generate_series(1,4500000) id;

-- table size
postgres=# \dt+ t
                                   List of relations
 Schema | Name | Type  |  Owner   | Persistence | Access method |  Size   | Description
--------+------+-------+----------+-------------+---------------+---------+-------------
 public | t    | table | postgres | permanent   | heap          | 1034 MB |
-- index size
postgres=# \di+ t_pkey
                                       List of relations
 Schema |  Name  | Type  |  Owner   | Table | Persistence | Access method | Size  | Description
--------+--------+-------+----------+-------+-------------+---------------+-------+-------------
 public | t_pkey | index | postgres | t     | permanent   | btree         | 96 MB |
```

1. Force seq scan

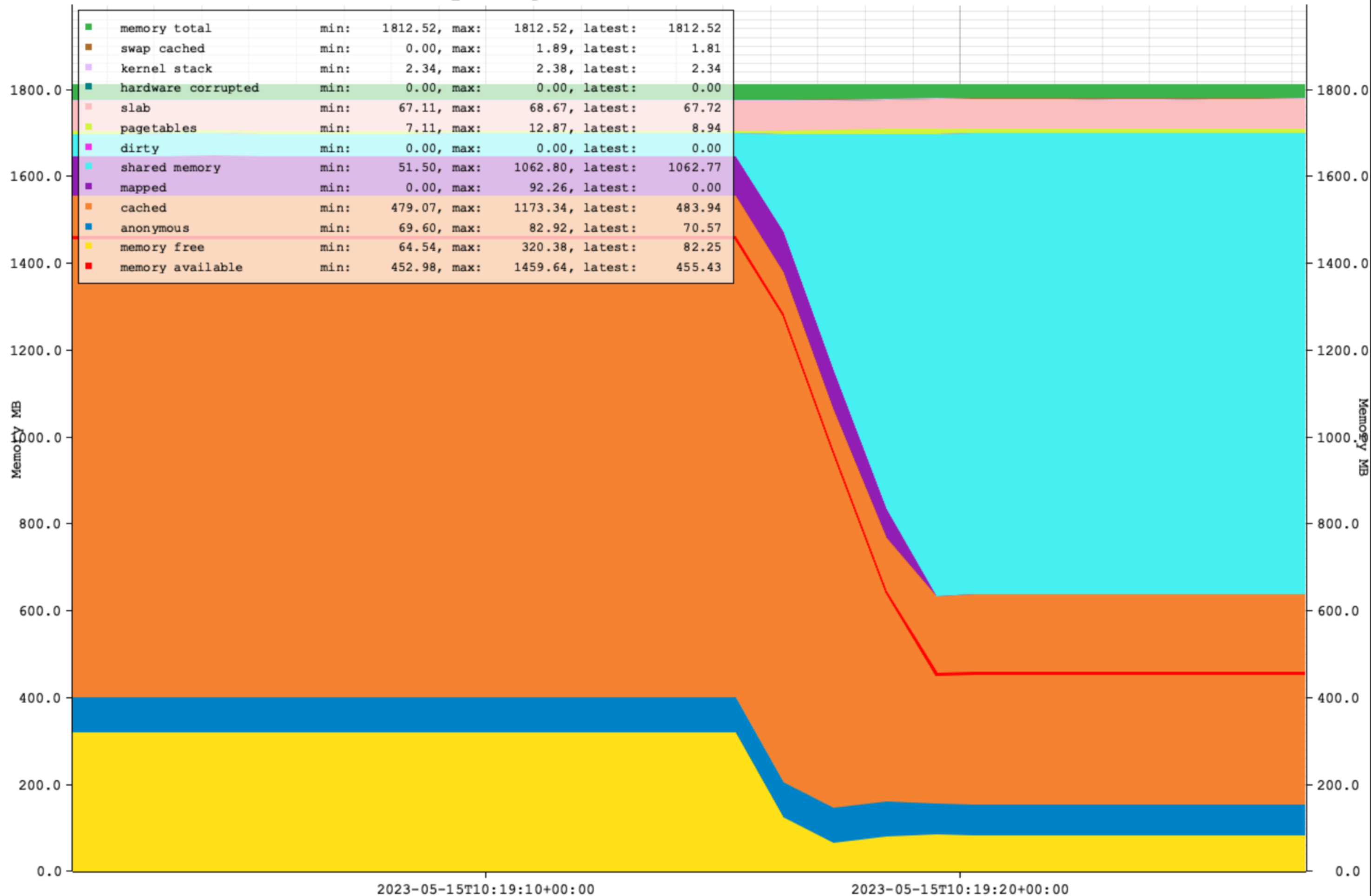
```
explain analyze select count(f1) from t;
                                                                 QUERY PLAN
------------------------------------------------------------------------------------------------
 Finalize Aggregate  (cost=156790.72..156790.73 rows=1 width=8) (actual time=2395.054..2403.280 rows=1 loo
   ->  Gather  (cost=156790.50..156790.71 rows=2 width=8) (actual time=2394.649..2403.272 rows=3 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Partial Aggregate  (cost=155790.50..155790.51 rows=1 width=8) (actual time=2367.083..2367.08
               ->  Parallel Seq Scan on t  (cost=0.00..151103.00 rows=1875000 width=200) (actual time=0.6
 Planning Time: 7.306 ms
 Execution Time: 2403.614 ms
(8 rows)
```



Memory usage: 192.168.66.100:9300:metrics
memory total min: 1812.52, max: 1812.52, latest: 1812.52
swap cached min: 0.00, max: 0.00, latest: 0.00
kernel stack min: 2.31, max: 2.38, latest: 2.34
hardware corrupted min: 0.00, max: 0.00, latest: 0.00
slab min: 65.89, max: 68.27, latest: 68.27
pagetables min: 6.89, max: 7.37, latest: 7.11
dirty min: 0.01, max: 0.02, latest: 0.02
shared memory min: 50.12, max: 51.67, latest: 51.50
mapped min: 89.66, max: 92.54, latest: 92.09
cached min: 118.48, max: 1153.55, latest: 1153.55
anonymous min: 78.19, max: 82.61, latest: 80.11
memory free min: 320.29, max: 1364.03, latest: 320.41
memory available min: 1457.43, max: 1464.58, latest: 1459.66
Memory MB
2023-05-15T10:17:40+00:00
2023-05-15T10:17:45+00:00
2023-05-15T10:17:50+00:00

2. Force index scan

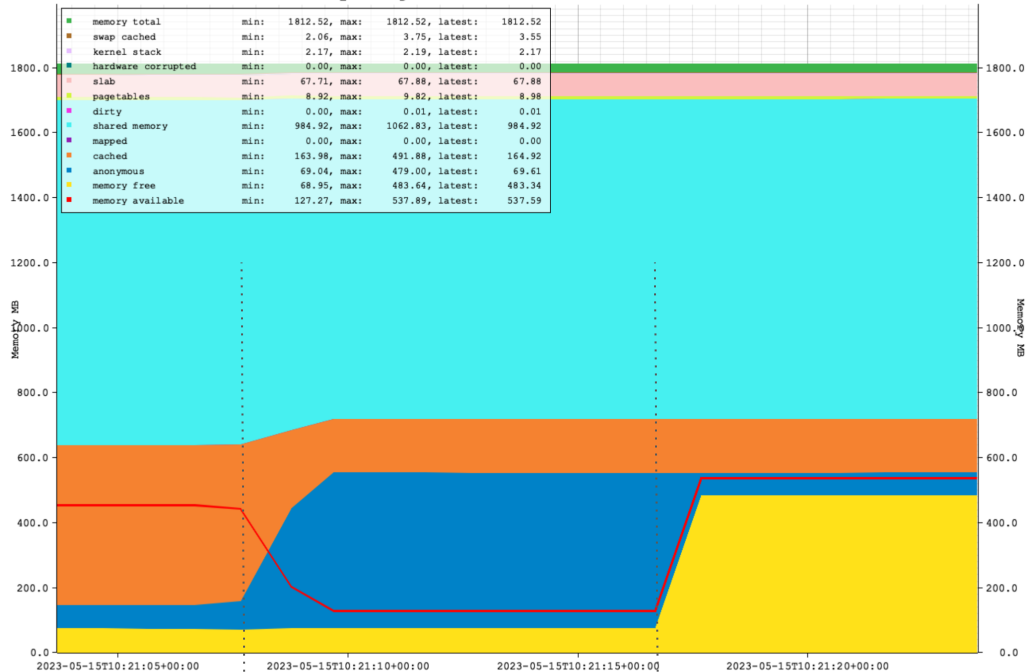
```
set enable_seqscan=off;
explain analyze select count(f1) from t where id > 0;
                                                          QUERY PLAN
-------------------------------------------------------------------------------------------------------
 Index Scan using t_pkey on t  (cost=0.43..260474.43 rows=4500000 width=204) (actual time=0.036..4414.006
   Index Cond: (id > 0)
 Planning Time: 6.661 ms
 Execution Time: 4716.215 ms
(4 rows)
```



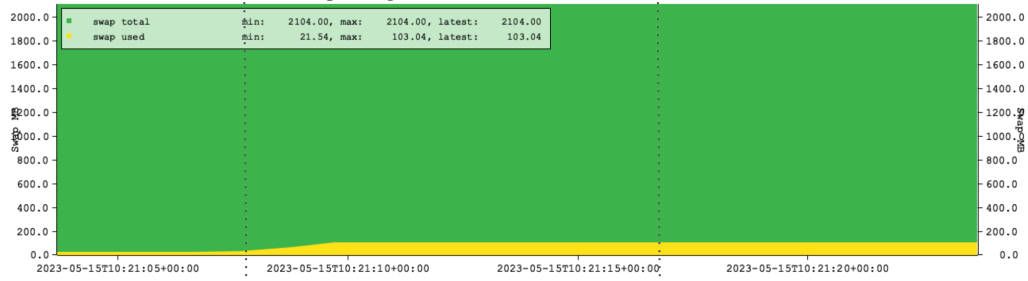
Memory usage: 192.168.66.100:9300:metrics
memory total min: 1812.52, max: 1812.52, latest: 1812.52
swap cached min: 0.00, max: 1.89, latest: 1.81
kernel stack min: 2.34, max: 2.38, latest: 2.34
hardware corrupted min: 0.00, max: 0.00, latest: 0.00
slab min: 67.11, max: 68.67, latest: 67.72
pagetables min: 7.11, max: 12.87, latest: 8.94
dirty min: 0.00, max: 0.00, latest: 0.00
shared memory min: 51.50, max: 1062.80, latest: 1062.77
mapped min: 0.00, max: 92.26, latest: 0.00
cached min: 479.07, max: 1173.34, latest: 483.94
anonymous min: 69.60, max: 82.92, latest: 70.57
memory free min: 64.54, max: 320.38, latest: 82.25
memory available min: 452.98, max: 1459.64, latest: 455.43
Memory MB
2023-05-15T10:19:10+00:00
2023-05-15T10:19:20+00:00

2. Force backend allocation

```
set my.count to 409600;
set my.size to 1020;
do
$$
  declare
    text_array text[];
    counter    int:= current_setting('my.count',true);
    size       int:= current_setting('my.size',true);
    size_read  varchar;
  begin
    raise info 'Pid: %', pg_backend_pid();
    raise info 'Array element size: %, count: %', size, counter;
    for count in 1..counter loop
      text_array[count]:=repeat('x',size);
    end loop;
    raise info 'done!';
    select pg_size_pretty(sum(total_bytes)) into size_read from pg_backend_memory_contexts;
    raise info 'size now: %', size_read;
    perform pg_sleep(60);
  end
$$;
```



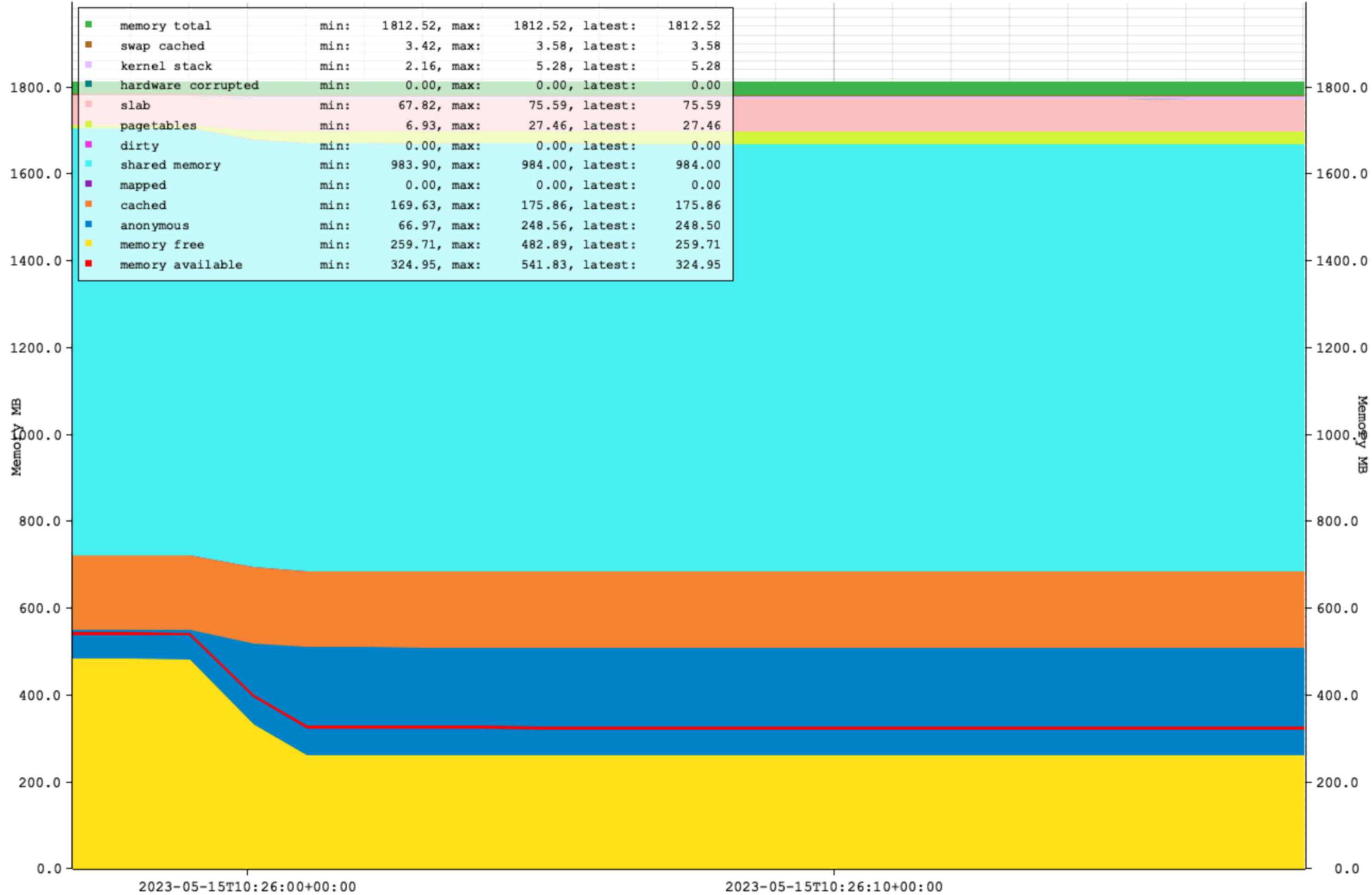
Memory usage: 192.168.66.100:9300:metrics
memory total min: 1812.52, max: 1812.52, latest: 1812.52
swap cached min: 2.06, max: 3.75, latest: 3.55
kernel stack min: 2.17, max: 2.19, latest: 2.17
hardware corrupted min: 0.00, max: 0.00, latest: 0.00
slab min: 67.71, max: 67.88, latest: 67.88
pagetables min: 8.92, max: 9.82, latest: 8.98
dirty min: 0.00, max: 0.01, latest: 0.01
shared memory min: 984.92, max: 1062.83, latest: 984.92
mapped min: 0.00, max: 0.00, latest: 0.00
cached min: 163.98, max: 491.88, latest: 164.92
anonymous min: 69.04, max: 479.00, latest: 69.61
memory free min: 68.95, max: 483.64, latest: 483.34
memory available min: 127.27, max: 537.89, latest: 537.59
Memory MB
2023-05-15T10:21:05+00:00
2023-05-15T10:21:10+00:00
2023-05-15T10:21:15+00:00
2023-05-15T10:21:20+00:00



Swap usage: 192.168.66.100:9300:metrics
swap total min: 2104.00, max: 2104.00, latest: 2104.00
swap used min: 21.54, max: 103.04, latest: 103.04
Swap MB
2023-05-15T10:21:05+00:00
2023-05-15T10:21:10+00:00
2023-05-15T10:21:15+00:00
2023-05-15T10:21:20+00:00

3.a Lots of backends -- idle

```
for T in $(seq 1 100); do
    psql -c "select pg_sleep(60)" &
done
```



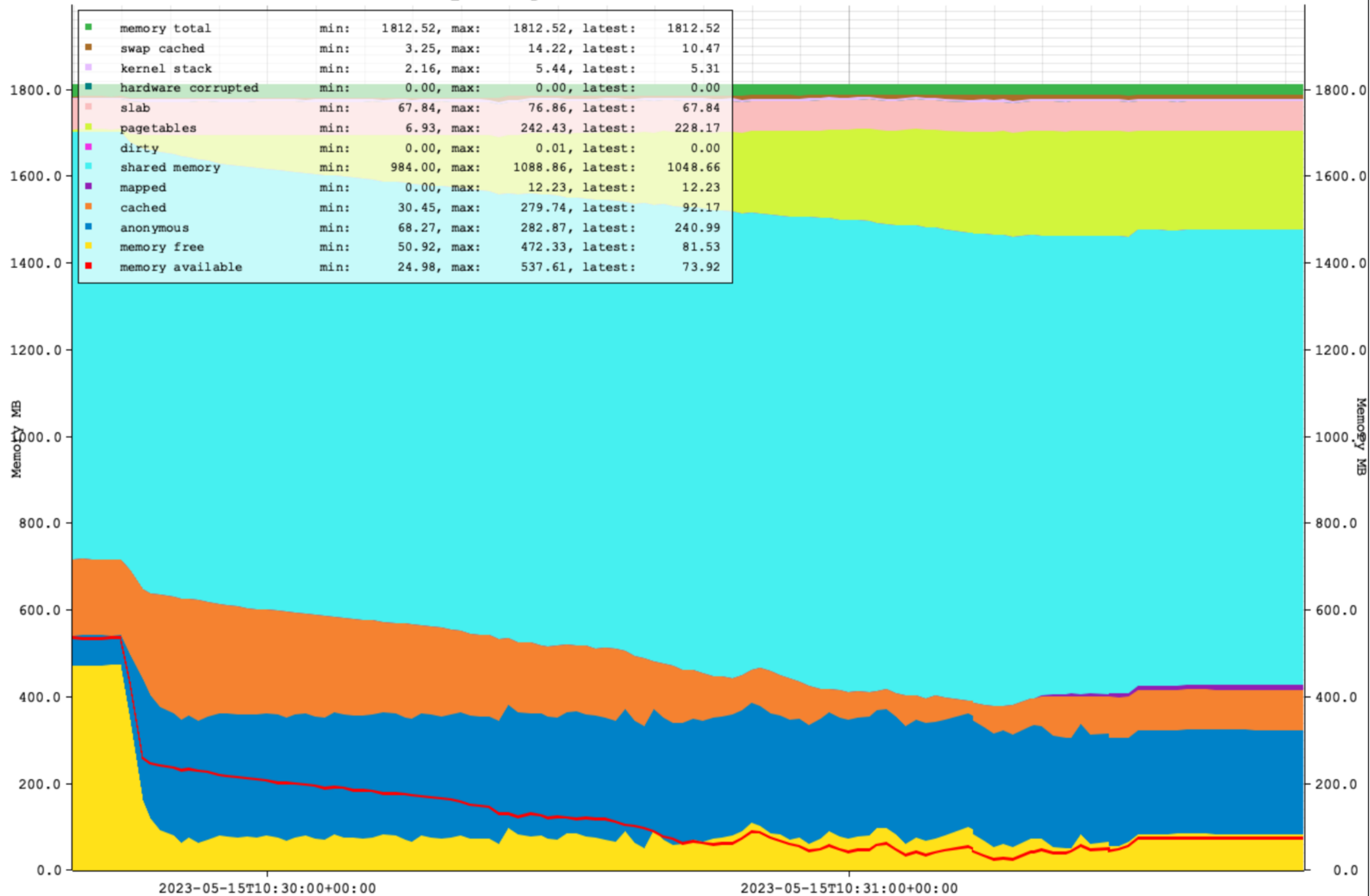
Memory usage: 192.168.66.100:9300:metrics
memory total min: 1812.52, max: 1812.52, latest: 1812.52
swap cached min: 3.42, max: 3.58, latest: 3.58
kernel stack min: 2.16, max: 5.28, latest: 5.28
hardware corrupted min: 0.00, max: 0.00, latest: 0.00
slab min: 67.82, max: 75.59, latest: 75.59
pagetables min: 6.93, max: 27.46, latest: 27.46
dirty min: 0.00, max: 0.00, latest: 0.00
shared memory min: 983.90, max: 984.00, latest: 984.00
mapped min: 0.00, max: 0.00, latest: 0.00
cached min: 169.63, max: 175.86, latest: 175.86
anonymous min: 66.97, max: 248.56, latest: 248.50
memory free min: 259.71, max: 482.89, latest: 259.71
memory available min: 324.95, max: 541.83, latest: 324.95
Memory MB
2023-05-15T10:26:00+00:00
2023-05-15T10:26:10+00:00

3.a Lots of backends -- idle

```
postgres=# select pg_size_pretty(sum(total_bytes)) from pg_backend_memory_contexts;
 pg_size_pretty
----------------
 1862 kB
```

3.b Lots of backends -- just doing simple things

```
for T in $(seq 1 100); do
    psql -c "set enable_seqscan=off; select count(f1) from t where id > 0; select pg_sleep(60)" &
done
```



Memory usage: 192.168.66.100:9300:metrics
memory total min: 1812.52, max: 1812.52, latest: 1812.52
swap cached min: 3.25, max: 14.22, latest: 10.47
kernel stack min: 2.16, max: 5.44, latest: 5.31
hardware corrupted min: 0.00, max: 0.00, latest: 0.00
slab min: 67.84, max: 76.86, latest: 67.84
pagetables min: 6.93, max: 242.43, latest: 228.17
dirty min: 0.00, max: 0.01, latest: 0.00
shared memory min: 984.00, max: 1088.86, latest: 1048.66
mapped min: 0.00, max: 12.23, latest: 12.23
cached min: 30.45, max: 279.74, latest: 92.17
anonymous min: 68.27, max: 282.87, latest: 240.99
memory free min: 50.92, max: 472.33, latest: 81.53
memory available min: 24.98, max: 537.61, latest: 73.92
Memory MB
2023-05-15T10:30:00+00:00
2023-05-15T10:31:00+00:00

4. Large backend allocations

Here is an oddity I found that could be very confusing.

In order to show this:

- I restart the server to decrease allocated shared memory.
- This allows anonymous allocations to be the majority of memory allocated.
- I run the PLpgSQL array allocation procedure with different settings:

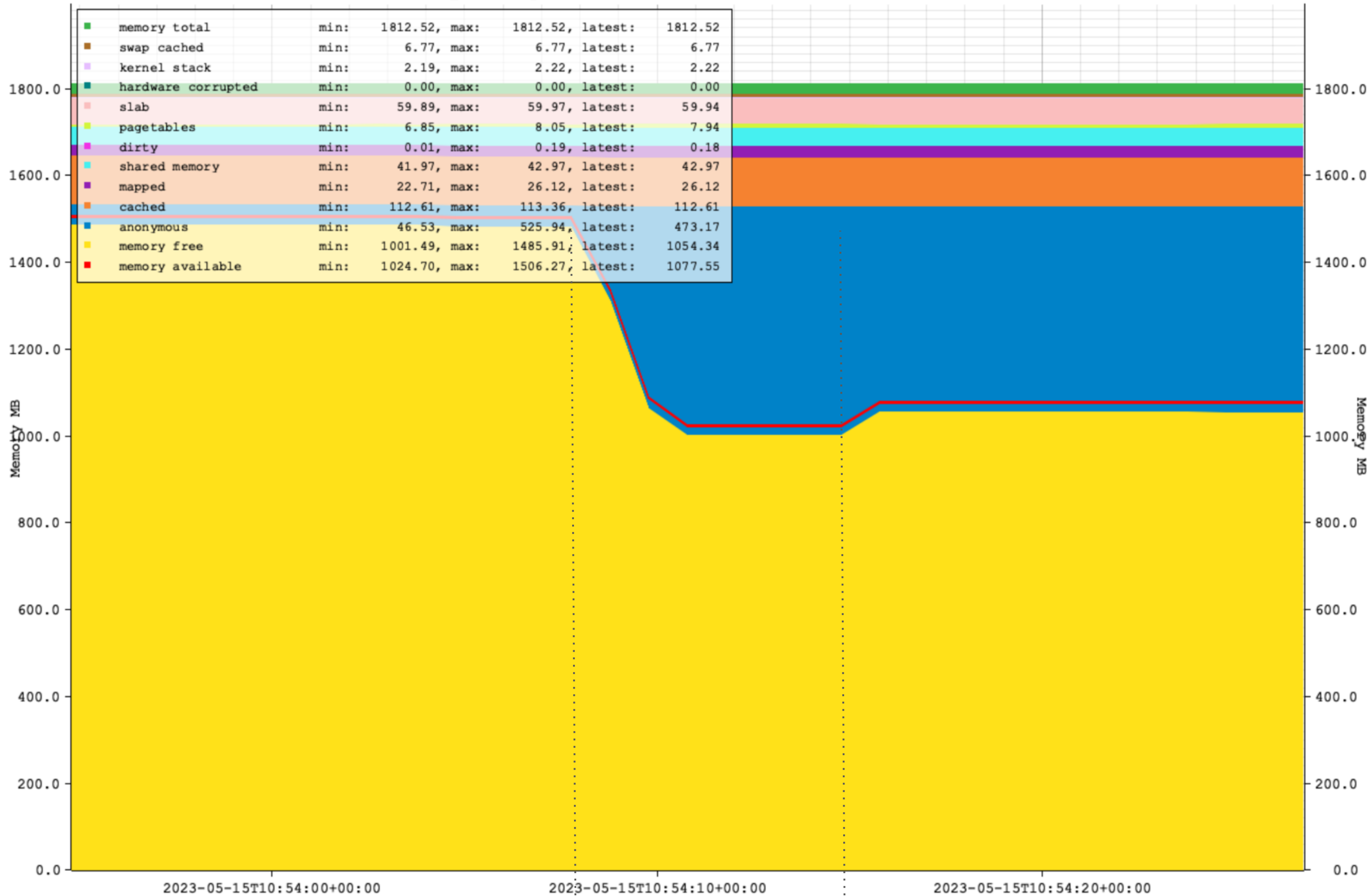
4. Large backend allocations

```
select pg_size_pretty(sum(total_bytes)) from pg_backend_memory_contexts;
set my.count to 3000;
set my.size to 166666;
do
$$
  declare
    text_array text[];
    counter    int:= current_setting('my.count',true);
    size       int:= current_setting('my.size',true);
    size_read  varchar;
  begin
    raise info 'Pid: %', pg_backend_pid();
    raise info 'Array element size: %, count: %', size, counter;
    for count in 1..counter loop
      text_array[count]:=repeat('x',size);
    end loop;
    raise info 'done!';
    select pg_size_pretty(sum(total_bytes)) into size_read from pg_backend_memory_contexts;
    raise info 'size now: %', size_read;
    perform pg_sleep(5);
  end
$$;
select pg_size_pretty(sum(total_bytes)) from pg_backend_memory_contexts;
```

4. Large backend allocations

```
postgres=# \i alloc_x.sql
 pg_size_pretty
----------------
 1755 kB
(1 row)

SET
SET
psql:alloc_x.sql:56: INFO:  Pid: 8646
psql:alloc_x.sql:56: INFO:  Array element size: 166666, count: 3000
psql:alloc_x.sql:56: INFO:  done!
psql:alloc_x.sql:56: INFO:  size now: 479 MB
DO
 pg_size_pretty
----------------
 1774 kB
(1 row)
```



Memory usage: 192.168.66.100:9300:metrics
memory total min: 1812.52, max: 1812.52, latest: 1812.52
swap cached min: 6.77, max: 6.77, latest: 6.77
kernel stack min: 2.19, max: 2.22, latest: 2.22
hardware corrupted min: 0.00, max: 0.00, latest: 0.00
slab min: 59.89, max: 59.97, latest: 59.94
pagetables min: 6.85, max: 8.05, latest: 7.94
dirty min: 0.01, max: 0.19, latest: 0.18
shared memory min: 41.97, max: 42.97, latest: 42.97
mapped min: 22.71, max: 26.12, latest: 26.12
cached min: 112.61, max: 113.36, latest: 112.61
anonymous min: 46.53, max: 525.94, latest: 473.17
memory free min: 1001.49, max: 1485.91, latest: 1054.34
memory available min: 1024.70, max: 1506.27, latest: 1077.55
Memory MB
0.0
200.0
400.0
600.0
800.0
1000.0
1200.0
1400.0
1600.0
1800.0
2023-05-15T10:54:00+00:00
2023-05-15T10:54:10+00:00
2023-05-15T10:54:20+00:00

4. Large backend allocations – short explanation

Anonymous allocations in PostgreSQL are performed by `palloc()`.
Palloc is a PostgreSQL abstraction, and calls the OS-level `malloc()` call.

malloc() is not a system call. It is a **library** to manage allocations in an address space.

So it's a memory management software layer between PostgreSQL and the OS.

It keeps allocated chunks around/cached 'in case you need it'.

- Small enough allocations are hardly visible/influencing.
- Very large allocations will get freed.
- Medium to large allocations are cached and can be significant.

Malloc resources

Malloc internals: https://sourceware.org/glibc/wiki/MallocInternals

Azeria heap exploitation part 1 (intro): https://azeria-labs.com/heap-exploitation-part-1-understanding-the-glibc-heap-im

Azeria heap exploitation part 2 (free): https://azeria-labs.com/heap-exploitation-part-2-glibc-heap-free-bins/

Conclusion

- Memory statistics in linux require interpretation and calculation.
    - I am not sure mine is 100% correct, but it seems to give a reasonable impression.
    - Any updates or corrections are welcome!

- Memory allocations in linux are lazy allocated.
    - And therefore PostgreSQL allocations are lazy allocated too.
    - Notably the buffer cache; anything else in the backend is even more dynamic.

- For architects and testers:
    - This means you need to build quite sophisticated test cases with real life data to be realistic.
    - Too simple test cases mean you actually/probably test a different scenario.

Conclusion

- Free memory != usable memory.
  - Look at available memory.
  - Too much free memory consistently reported likely means not using all memory.

- Main common PostgreSQL allocations:
  - Shared memory (== mapped memory == cached memory): buffer cache.
  - Anonymous memory: backend memory allocation.

- PostgreSQL is exhaustively documented to be needing OS cache.
  - Actual cache == cached - shared memory - mapped memory - dirty pages
  - Do you leave memory for OS caching?
    - If not: this will make query latency fluctuate a lot for seemingly identical situations.

Conclusion

- Large seq scans will only use 256k of buffers as ring.
  - Large means > 25% of set buffer cache size.
- Backends require a low amount of memory.
  - But need to build a pagetables to access buffer cache.
  - Any actual work requiring backend memory allocation comes on top.
  - Therefore: be very reserved about the number of backends.
- Malloc can play memory tricks.
  - For small allocations this is probably as good as invisible.
  - For 'medium' allocations this might play up.
  - I am not sure how often this plays up, but it's hard to spot if you're unaware.

Conclusion

- dsar (which created the memory graphs):
  - https://github.com/fritshoogland-yugabyte/dsar
  - Any requests are welcome!