>>

PostgreSQL Buffer Manager

^ >>

❖ PostgreSQL Buffer Manager

PostgreSQL buffer manager:

- provides a shared pool of memory buffers for all backends
- all access methods get data from disk via buffer manager

Buffers are located in a large region of shared memory.

Definitions: **`src/include/storage/buf*.h`**

Functions: **`src/backend/storage/buffer/*.c`**

Buffer code is also used by backends who want a private buffer pool

<< ^ >>

❖ PostgreSQL Buffer Manager (cont)

Buffer pool consists of:

`BufferDescriptors`

- shared fixed array (size **`NBuffers`**) of **`BufferDesc`**

`BufferBlocks`

- shared fixed array (size **`NBuffers`**) of 8KB frames

`Buffer` = index values in above arrays

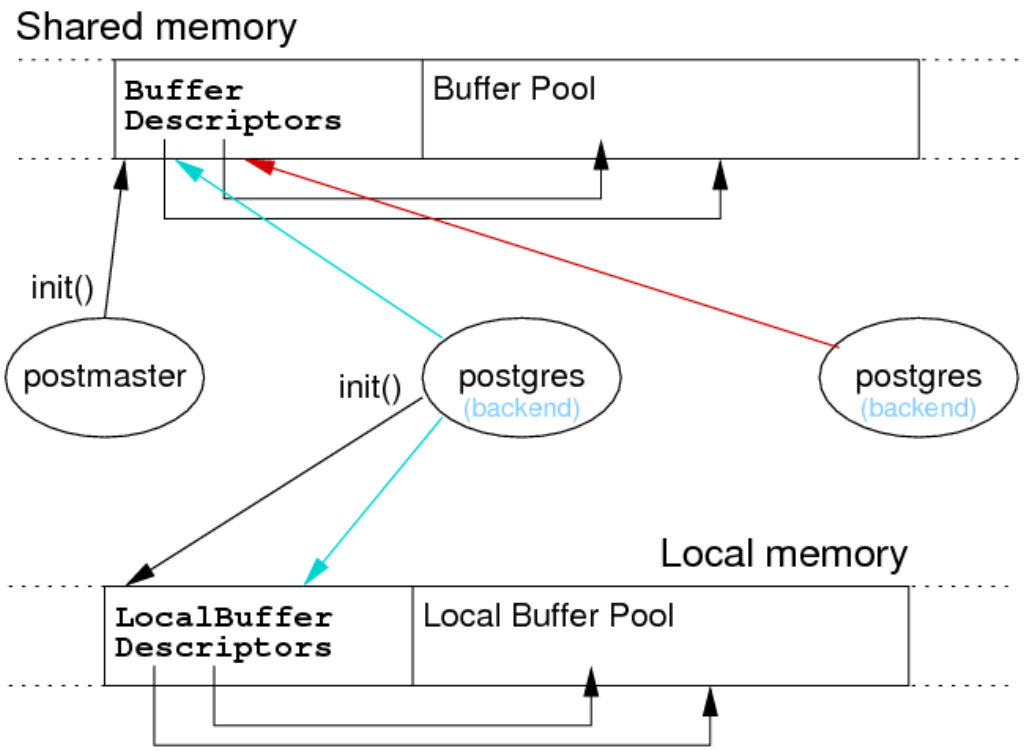
- indexes: global buffers **`1..NBuffers`**; local buffers negative

Size of buffer pool is set in postgresql.conf, e.g.

```
shared_buffers = 16MB    # min 128KB, 16*8KB buffers
```

<< ^ >>

❖ PostgreSQL Buffer Manager (cont)



COMP9315 21T1 ♢ PG Buffers ♢ [3/8]

<< ^ >>

❖ PostgreSQL Buffer Manager (cont)

include/storage/buf.h

- basic buffer manager data types (e.g. **Buffer**)

include/storage/bufmgr.h

- definitions for buffer manager function interface
  (i.e. functions that other parts of the system call to use buffer manager)

include/storage/buf_internals.h

- definitions for buffer manager internals (e.g. **BufferDesc**)

Code: **backend/storage/buffer/*.c**

Commentary: **backend/storage/buffer/README**

❖ PostgreSQL Buffer Manager (cont)

Definition of buffer descriptors simplified:

```
typedef struct BufferDesc
{
    BufferTag  tag;      // ID of page contained in buffer
    int        buf_id;   // buffer's index number (from 0)

    // state, containing flags, refcount and usagecount
    pg_atomic_uint32 state;

    int        freeNext;  // link in freelist chain
    ...
} BufferDesc;
```

<< ^ >>

❖ Clock-sweep Replacement Strategy

PostgreSQL page replacement strategy: clock-sweep

- treat buffer pool as circular list of buffer slots
- **`NextVictimBuffer`** (NVB) holds index of next possible evictee
- if **`Buf[NVB]`** page is pinned or "popular", leave it
  - **`usage_count`** implements "popularity/recency" measure
  - incremented on each access to buffer (up to small limit)
  - decremented each time considered for eviction
- else if **`pin_count`** = 0 and **`usage_count`** = 0 then grab this buffer
- increment **`NextVictimBuffer`** and try again (wrap at end)

<< ^ >>

❖ Clock-sweep Replacement Strategy (cont)

Action of clock-sweep:

NVB

	[0]	[1]	[2]	[3]	[4]	[5]
Before Clock Sweep	pin: 1 use: 3	pin: 0 use: 1	pin: 0 use: 2	pin: 0 use: 0	pin: 1 use: 1	pin: 1 use: 2

NVB

	[0]	[1]	[2]	[3]	[4]	[5]
After Clock Sweep	pin: 1 use: 2	pin: 0 use: 0	pin: 0 use: 1	pin: 1 use: 0	pin: 1 use: 1	pin: 1 use: 2

<< ^

❖ Clock-sweep Replacement Strategy (cont)

For specialised kinds of access (e.g. sequential scan),

- clock-sweep is not the best replacement strategy
- can allocate a private "buffer ring"
- use this buffer ring with alternative replacement strategy

Produced: 22 Feb 2021