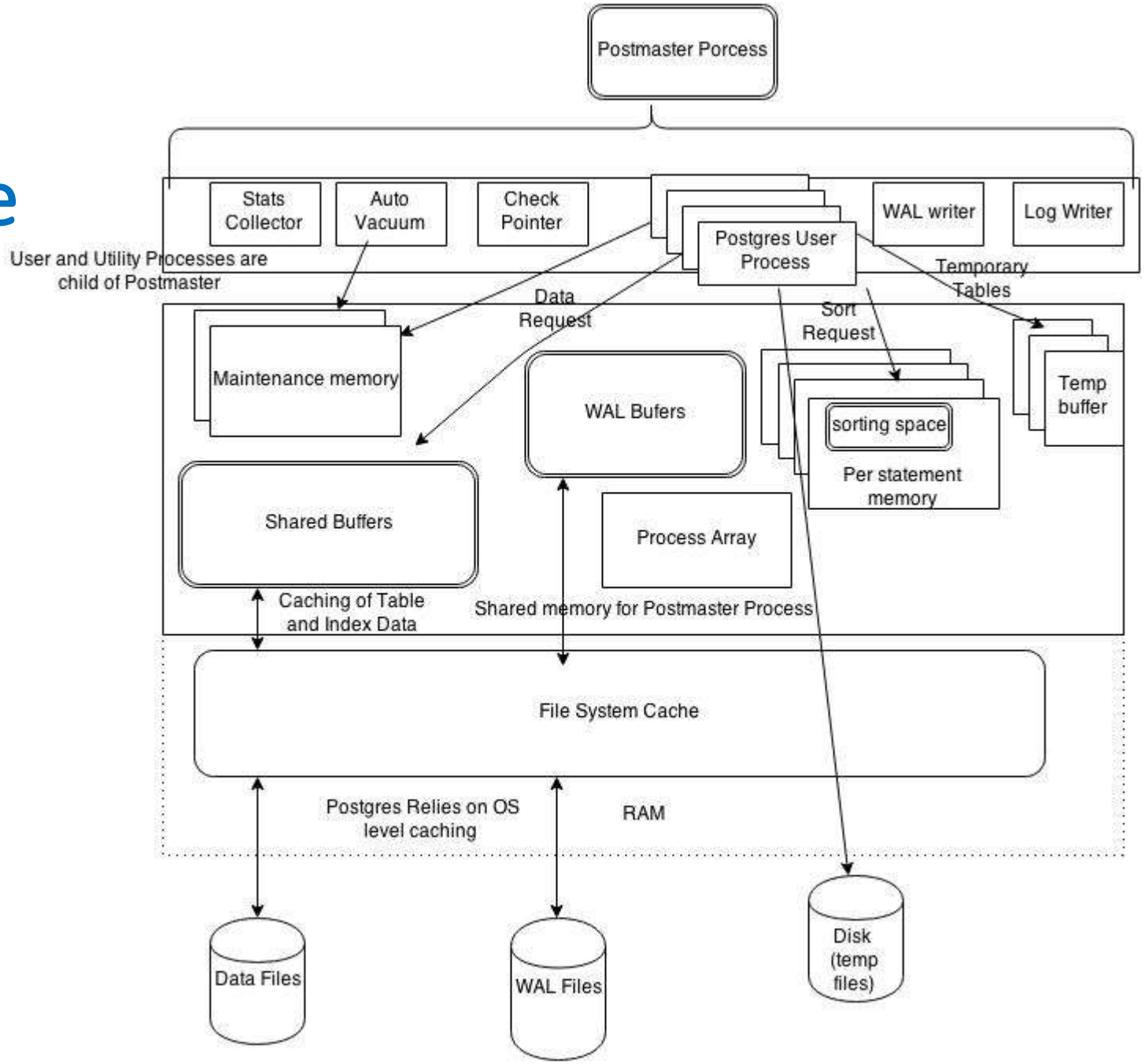
PostgreSQL Parameter Tuning-I Memory and Optimizer Parameters

ASHNIK

PostgreSQL Architecture



postgresql.conf

- Generally located inside data directory of the cluster
- Each line is a holds one parameter in ‘key-value’ pair separated by ‘=‘
- You can set parameters on command line on startup
- Supports include directive to include parameters from additional files

Other places to edit parameters

- Make changes at Database level for cluster hosting different databases of different nature
- Make changes at user level according to nature of job e.g. OLTP, Reports, Batches etc
- Make changes at session level/transaction level before an exceptionally costly/different query
- Note: Some parameters can only be changed on startup or in postgresql.conf

Memory Parameters

ASHNIK

Shared Buffer- shared_buffers

- Default value is really small- 32MB
- A Good Value:
  - Usually for better performance set this to 15%-30% of total available RAM on a dedicated DB server
  - On shared servers keep it large enough to accommodate indexes
- Remember:
  - On 32-bit setup a value above 2GB is not practical
- Tip:
  - On v9.2 and earlier set SHMMAX to proper value before setting shared_buffer

Sorting Memory – work_mem

- A higher value improves query performance faster with in-memory sort
- You can log [log_temp_files] the sorts which spills over to disk or see them in explain plan [e.g. Sort Method: external merge Disk: XkB]
- A good value:
  - Either the whole sort can be performed in memory [Sort Method: quick sort]
  - at least the intermediate result can be stored in memory [xKB in our case]
- Remember:
  - This is 'per sorting operation'
  - If a statement has multiple sort requirements those many sorting space will be allocated
- Tip:
  - Set the global/instance level value appropriately
  - Change values at Database, User or Session/Transaction level for specific operations

Memory for Temporary Objects– temp_buffer

- Amount of memory used for each session for caching the temporary tables
- **A good Value:**
  - Change it at session level Just before creating the first temporary table
- **Remember:**
  - A session will allocate temporary buffers as needed
  - About 64bytes to 8kb of over head is incurred for allocation/increment when session uses temporary buffer
- **Tip:**
  - Don’t change if you don’t use temporary tables

Maintenance Memory– maintenance_work_mem

- Helps improve the performance of maintenance operations e.g.
  - VACUUM - CREATE INDEX - ALTER TABLE ADD FOREIGN KEY
- <u>A good Value:</u>
  - Set this to a larger value than work_mem- significant to hold large tables for INDEXING and VACUUM operations
  - e.g. on a 64GB of RAM system, 2GB for maintenance_work_mem could be safe
- <u>Remember:</u>
  - A sessions can do only one maintenance operation at a time
  - Generally not many maintenance operations would be done in parallel
- <u>Tip:</u>
  - Parallel threads for autovacuum (max_autovacuum_worker) will consume multiple slots of maintenance memory

Query Planner Parameters

ASHNIK

Important Note

- These parameters are input to your optimizer
- These parameters help optimizer decide on the best out of all available execution plans
- These parameters do not
  - decide the actual performance
  - Define disk speed
  - allocate memory or disk space

Cost of Accessing a random page on Disk- random_page_cost

- Default- 4.0
- A higher value is more likely to push the optimizer to use a table scan (presumably sequential fetch of pages)
- A Good Value:
  - If you have faster disks set this to smaller values e.g. 2.0 or 3.0
  - For flash disk you may infact set to even lower values 1.5
- Remember:
  - This parameter does not define how fast Postgres 'will' access random pages
  - It defines how fast Postgres 'can expect' the request for random pages to be fulfilled
- Tip:
  - Check your explain plan and effective_cache_size (to be discussed) before setting/changing this parameter

Cost of Accessing a page sequentially on Disk- sequential_page_cost

- Default- 1.0
- You may want to reduce this value to account for caching effect
- A Good Value:
  - If you have faster disks (flash disk or SDD) set this to smaller values e.g. 0.8 or 0.9
- Remember:
  - This parameter does not define how fast Postgres ‘will’ access sequential pages
  - It defines how fast Postgres ‘can expect’ the request for sequential pages to be fulfilled
- Tip:
  - You must always keep this lower than random_page_cost

Memory available to OS for File system cache – effective_cache_size

- Default- 128MB
- It helps optimizer envisage how much of memory is available for caching files specially while considering index scans
- A Good Value:
  - 50-80% of your RAM on a dedicated DB server is a good value
  - On a shared server estimate this value by looking at OS statistics for Free+Cached memory
- Remember:
  - This parameter is not allocation of memory for Postgres it is only indication/input of a n estimate to Postgres
- Tip:
  - A higher value is more likely to promote index scans

Force Plans with enable_* parameters

- Postgres has certain enable_* parameters
- These parameters help the optimizer include (if on) or exclude (if off) certain optimization techniques e.g.
  - Index utilization - Index only scans, Index scans, Bitmap scans
  - Joins- merge join, nested table loop join, hash join
- A Good Value:
  - Its wise to leave them to default
- Remember:
  - These are not hints! If you disabling a parameter will not use that optimization technique and *not 'just discourage it'* [enable_materialization is exception]
- Tip:
  - Change these only for specific queries or users in specific cases

/ashnikbiz

If you have more questions, write to us at:
success@ashnik.com

Website: www.ashnik.com

Thank you