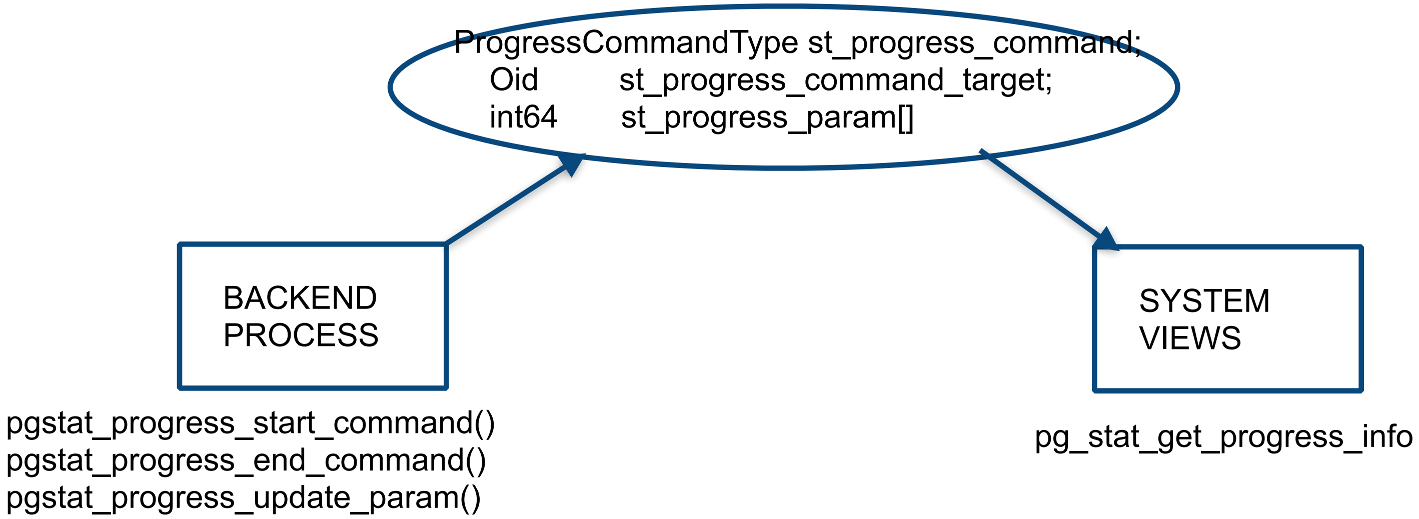
PostreSQL Monitoring Enhancements

-Rahila Syed

Progress Reporting API

- Progress reporting of utility commands
- Parameters reported
    - 10 64-bit counters in shared memory
    - OID of the relation which command targets
    - Type of the command for which progress is being reported
- System views examining the parameters reported


ProgressCommandType st_progress_command;
Oid st_progress_command_target;
int64 st_progress_param[]
BACKEND PROCESS
SYSTEM VIEWS
pgstat_progress_start_command()
pgstat_progress_end_command()
pgstat_progress_update_param()
pg_stat_get_progress_info

VACUUM Progress Checker

- System view - pg_stat_progress_vacuum
- Phases of VACUUM
  - Heap Scanning
  - Heap Vacuuming
  - Index Vacuuming
  - Cleaning up indexes
  - Truncating heap
  - Performing final cleanup
- Progress parameters

```
postgres=# select * from pg_stat_progress_vacuum;

 pid | datid | datname | relid | phase | heap_blks_total | heap_blks_scanned |
-----+-------+---------+-------+-------+-----------------+-------------------+-----------------
heap_blks_vacuumed | index_vacuum_count | max_dead_tuples | num_dead_tuples
—-——————————-+———————————+——————————-+——————————
(0 rows)
```

Progress Reporting in Other Databases

- Oracle
  - v$session_longops- Dynamic performance view
  - Track query running longer than 6 seconds
  - Commands
    - Table scan
    - Index Fast Full Scan
    - Hash join
    - Sort/Merge
  - Phases
    - Progress reports in phases of linear progress
    - Time remaining = elapsed_seconds * (totalwork - sofar)/sofar
  - Information

```
select opname, target, sofar, totalwork, units, elapsed_seconds, message from v$session_longops order by start_time desc;
```

Progress Reporting in Other Databases

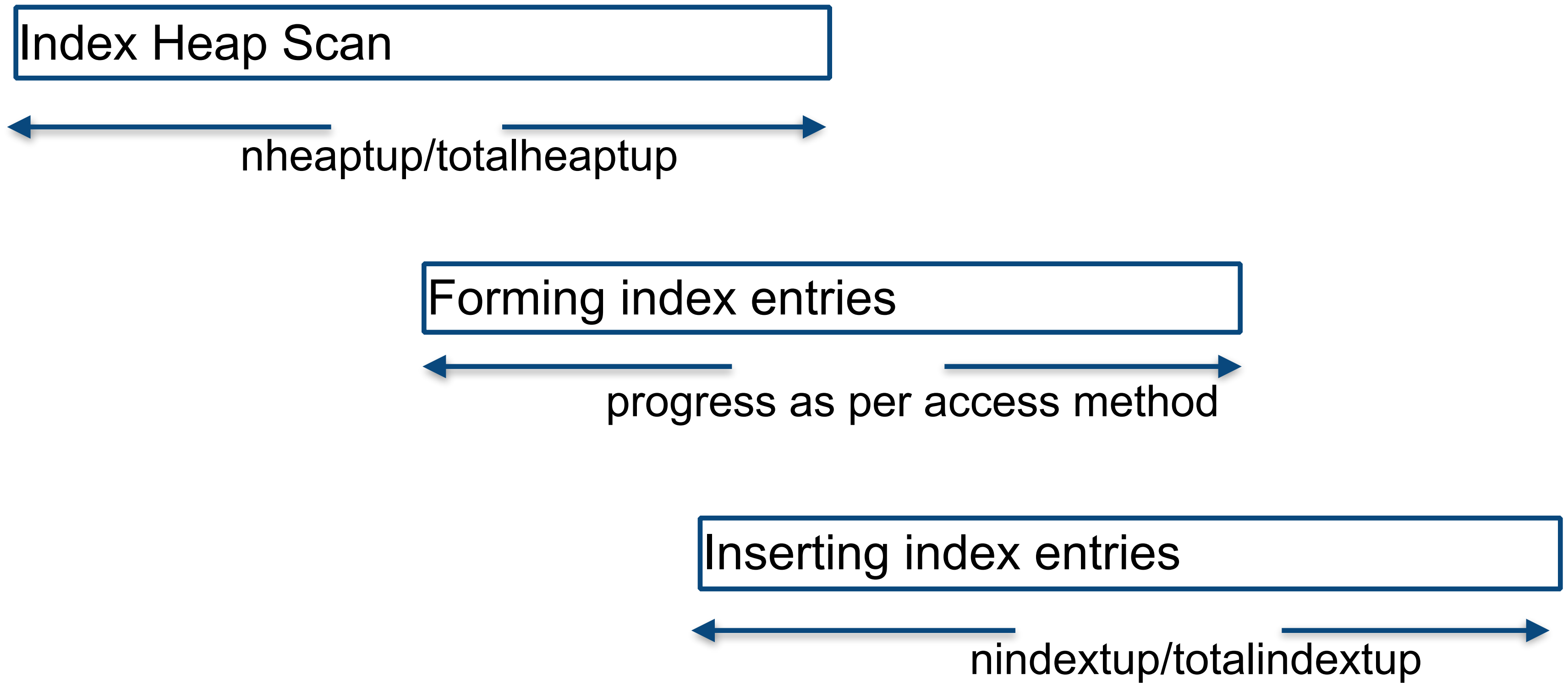
- MariaDB
  - Separate progress reporting for stages of the command
  - Commands
    - Alter table
    - Create index
    - Drop index
    - Load data infile
  - Information
    - Stage
    - Max_stage
    - Progress (within current stage)

```
ALTER TABLE my_mail ENGINE=maria;
Stage: 1 of 2 'copy to tmp table'  5.37% of stage done
```

Take Aways

- Progress is reported in phases
- The linear prediction can be wrong
- Report current state of operations

Create Index



Btree

- Phase 1 : Scanning the heap for tuples to be indexed
  - Number of tuples scanned versus total number of tuples
- Phase 2: Sorting the tuples
  - In memory sort : Fast and lesser need for a progress report
  - External merge sort: Multiple levels for reporting progress
- Phase 3: Write to the index
  - Number of tuples written versus total number of index tuples
- Phase 4: Writing statistics information
  - Updating heap and index pg_class rows

External Merge Sort

- Sort the batches of tuples that fit in memory and write to tapes as individual runs
  - The progress can be measured in terms heap blocks written versus total heap blocks in relation
- Tapes with sorted runs are merged
  - Compare the first runs on each tape writing the smallest tuple to an output tape.
  - The progress of this phase can be measured by counting the tuples written to output tape versus total index tuples.
- Polyphase merge
  - Each run is written once to tape for each pass
  - Progress can be measured by number of runs written to tapes versus total runs * number of passes

GIN

- Phase 1: Scan the heap for heap tuples to be indexed
    - Number of heap tuples versus total heap tuples
- Phase 2: Extract index entries from each heap tuple
    - Insert the index entries in temporary buffer, if the memory is full perform phase 3.
    - Number of heap tuples versus total heap tuples
- Phase 3: Insert remaining index entries from temporary buffer into an index
    - Index entries inserted versus total index entries
- Phase 4: Writing statistics information
- Phase 5: Writing WAL record

Gist

- Phase 1: Scan the heap for tuples to be indexed.
- Phase 2: Form the indexed tuple for each heap tuple
- Phase 3: Write the tuples to index
    - Number of heap tuples processed / total number of heap tuples
- Heap scan and index write has one to one mapping, as there is one entry per heap tuple
- Phase 4: Writing statistics information
- Phase 5: Writing WAL record

BRIN

- Phase 1: Scan the heap for tuples to be indexed
- Phase 2: Form one index tuple for each range of the blocks
  - Number of index entries = size of relation in pages / pages_per_range
- Phase 3: Write the tuple to index
  - Entries created till now / number of index entries.
- Overall progress can be measured by heap tuples scanned / total number of heap tuples
- Phase 4: Writing statistics information
- Phase 5: Writing WAL record

CREATE INDEX

- The parameters that can be reported for a create index
  - Oid of the target
  - Type of index
  - heap_tuples_scanned
  - total_heap_tuples
  - Type of sort (if applicable)
  - heap_blocks_sorted(if applicable)
  - index_tuples_inserted
  - total_index_tuples

CREATE INDEX

- Take away here is that different phases of an index scan can overlap
- In which case, it is will not return accurate estimate of remaining time
- Reporting progress of individual phases is the way to go
- Some times the individual phases are tightly coupled
- In which case it is fine to report progress in terms of one of the phases, like gist
- Progress measurement can be reasonably accurate if divided into linear phases

CLUSTER

- Phase1 : Scan the heap (either in index order or sequentially)
- Phase 2: Writing clustered table to new heap
  - Index scan : Each tuple scanned is immediately rewritten to new heap.
    - Progress can be reported as tuples scanned/rewritten versus total tuples in heap
  - Sequential scan : Tuple is first written to tuplesort memory.
    - Progress in this phase will be number of tuples accumulated for sorting against total number of tuples in the heap.
    - Tuples sorting
      - Progress of this phase can me measured similar to progress of external merge sort.
    - Sorted tuples obtained are written into the heap.
      - Progress of this phase can be measured as tuples written against total number of tuples.
- Phase 3: Swap relation files

ALTER TABLE

- Phase 1: Permission checks , preliminary examination, creation of work queues
- Phase 2: Executing the list of commands to be applied to the table
  - Divided into multiple passes for subcommands
  - Builds an index if phase 3 does not exist
- Phase 3: Check new constraints and rewrite the table/indexes
  - Report tuples scanned versus total tuples in the table
  - Progress of rewrite index relations same as create index

Wait Events

- Wait events are the events that occur during a database operation when a request has to be processed
- Current wait events infrastructure reports information about the type of the wait event a backend is waiting on at that instant.
- It gives information on which event the query is waiting on if any at particular instant of time
- Although to derive information about the bottlenecks in the system historic data needs to be gathered
- This can be achieved by sampling the wait event info from pg_stat_activity over certain intervals of time.

Wait Events

- Wait events report where the backend is waiting
- Initially events reported were heavyweight , lightweight lock information
- Two columns in pg_stat_activity
  - wait_event_type
  - wait event
- Recent advancements include wait events for
  - Activity
  - Client
  - Extension
  - IPC
  - Timeout
  - I/O

THANK YOU!

ANY QUESTIONS?

email id: rahila.syed@enterprisedb.com/ rahilasyed.90@gmail.com