Database - Feature #4980

Investigate the impact of TOAST manipulation in PotgreSQL engine

10/23/2020 11:22 AM - Adrian Lungu

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

PostgreSQL has a manager (named TOAST) which ensures that all texts wider than TOAST_TUPLE_THRESHOLD (usually 2kB) are not stored inline, but separately, so that queries won't have to read the whole huge row. There are TOAST settings for each column (for example: EXTENDED allows compression and eventually separate storage, while PLAIN inhibits such management). PostgreSQL marks all text columns as EXTENDED, so the TOAST manager can deal with eventual big texts.

The problem: Lets consider two type of tables, one with medium-sized texts (almost 2kB) and one with large-sized texts (over 2 kB). A full table scan (or a big indexed scan) which retrieves only the ids will be slower for medium-sized texts than for large-sized texts. This is because, on a low level, the queries read the whole row. The medium-sized texts are read entirely, while the large-sized texts are not (because they are TOASTed, so they are not inline). This difference can be huge (for example 3198ms for medium-sized vs 85ms for large-sized).

A possible solution: An initial solution is to decrease TOAST_TUPLE_THRESHOLD (eventually to 128B) so that medium-sized texts will be forced to be stored separately, so queries which shouldn't read the text field are way more quick. The manipulation of TOAST_TUPLE_THRESHOLD is enabled starting with PostgreSQL 11.

The disadvantage: Storing so many texts out of line will lead to a slower query of them. Also, compressing them will lead to slower operations involving sub-strings (like CONTAINS). However, the EXTERNAL strategy can inhibit the compression of strings.

Preliminary sub-tasks:

- Set-up a PostgreSQL >= 11 cluster (according to the documentation this is the first version which allows TOAST_TUPLE_THRESHOLD setting) and find a scenario which reproduces the time difference above. Try to find out how much the time consumption is decreased by modifying the threshold.
- Experiment with different strategies: MAIN, PLAIN, EXTENDED, EXTERNAL.
- Try out scenarios in which sub-string operations are used (like CONTAINS) in order to detect the trade-off between encrypted and not-encrypted storage.
- Try out scenarios in which the out-of-line texts are queries in order to detect the trade-off between in-line and out-of-line.