CSS3 vs CSS4 - what are the benefits of upgrading

Summary: documentation about the benefits of upgrading to CAST Storage Service 4, from CAST Storage Service 3.

Introduction

CAST Storage Service (CSS) 3 is a re-packaged **PostgreSQL 9.6.11**, while **CAST Storage Service 4** uses **PostgreSQL 13**. Since the **EOL** for PostgreSQL 9.6.x is scheduled for **November 2021**, CAST would encourage you to upgrade your instance wherever possible. Over and above the benefits of using a PostgreSQL release that is not due to go **EOL** until November 2025, the **improvements** added to PostgreSQL between 9.6 and 13 are worth highlighting - these improvements also provide better performance for use with CAST AIP. The major benefits are listed below.

Note that CAST routinely recommends using PostgreSQL on Linux, rather than CAST Storage Service on Microsoft Windows since performance is always better.

Schema size on disk

The default CAST AIP schemas created on a CSS4/PostgreSQL 13 instance are **smaller in size** when compared to the same schemas created on a CSS3 /PostgreSQL 9.6.x instance. This is primarily due to the fact that index storage handling has improved in PostgreSQL 13: the B-tree index takes up less space on disk.

Faster VACUUM

The built in **VACUUM process is much faster** on CSS4/PostgreSQL 13 because of the ability to vacuum indexes in parallel which was not supported in CSS3/PostgreSQL 9.6.x (you can run a VACUUM process directly in AIP Console - see **Administration Center - Settings - CSS Optimization**). If you are running VACUUM manually using PgAdmin or an equivalent tool, you can also specify the PARALLEL directive to control how many parallel VACUUM jobs you want running on your indexes - see **https://www.postgresql.org/docs/current/sql-vacuum.html**.

Query performance improvement

Query performance has been improved in CSS4/PostgreSQL 13 because incremental sort has been implemented for multi-column sorts. This performance improvement is easily highlighted using the following query executed on both CSS4/PostgreSQL 13 and CSS3/PostgreSQL 9.6:

```
DROP TABLE IF EXISTS abc;
CREATE TABLE abc (x int PRIMARY KEY, y int);
CREATE INDEX ON abc (y);
INSERT INTO abc
SELECT x, x % 16378
FROM generate_series(1,1000000) x;

SELECT pg_size_pretty(pg_relation_size('abc_y_idx')); -- Get the size of index

EXPLAIN ANALYZE
SELECT count(*)
FROM abc
WHERE y = 50;

DROP INDEX abc_y_idx;
ANALYZE abc;

EXPLAIN ANALYZE
SELECT *
FROM abc
WHERE x % 13 = 0
ORDER BY x, y DESC
LIMIT 13;
```

The query plan for CSS3/PostgreSQL 9.6 and CSS4/PostgreSQL 13 are shown below - the **execution time** shown on the last line is vastly better when using CSS4/PostgreSQL 13:

CSS3/PostgreSQL 9.6 Query Plan

```
Limit  (cost=19542.51..19542.54 rows=13 width=8) (actual time=373.912..373.916 rows=13 loops=1)
  ->  Sort  (cost=19542.51..19555.01 rows=5000 width=8) (actual time=373.910..373.912 rows=13 loops=1)
        Sort Key: x, y DESC
        Sort Method: top-N heapsort  Memory: 25kB
        ->  Seq Scan on abc  (cost=0.00..19425.00 rows=5000 width=8) (actual time=0.061..359.229 rows=76923
loops=1)
              Filter: ((x % 13) = 0)
              Rows Removed by Filter: 923077
Planning time: 0.839 ms
Execution time: 373.971 ms
```

CSS4/PostgreSQL 13 Query Plan

```
Limit  (cost=7.52..100.14 rows=13 width=8) (actual time=0.135..0.140 rows=13 loops=1)
  ->  Incremental Sort  (cost=7.52..35633.43 rows=5000 width=8) (actual time=0.133..0.136 rows=13 loops=1)
        Sort Key: x, y DESC
        Presorted Key: x
        Full-sort Groups: 1  Sort Method: quicksort  Average Memory: 25kB  Peak Memory: 25kB
        ->  Index Scan using abc_pkey on abc  (cost=0.42..35408.43 rows=5000 width=8) (actual time=0.044..0.113
rows=14 loops=1)
              Filter: ((x % 13) = 0)
              Rows Removed by Filter: 168
Planning Time: 0.617 ms
Execution Time: 0.177 ms
```

Overall analysis/snapshot execution time improvements

Running the same analysis/snapshot on the same host server using CSS3 and CSS4 shows an improvement in performance for CSS4:

CSS3 (12 GB RAM, 4 CPU)	CSS4 (12 GB RAM, 4 CPU)
totalAnalysisDuration = 02:46:34.368	totalAnalysisDuration = 02:41:40.424
totalSnapshotDuration = 00:19:43.032	totalSnapshotDuration = 00:15:53.693