HP / February 12, 2010 12:16AM

Performance Issue in PostgreSQL: Counting rows in a table

In summary, PostgreSQL sacrifices some performance to the overall reliability and data integrity. COUNT is slow for the whole table scan but with normal speed when COUNT with INDEXed tables.

--

The article below is from Why PostgreSQL Instead of MySQL 2009.

--

Counting rows in a table

One operation that PostgreSQL is known to be slow performing is doing a full count of rows in a table, typically using this SQL:

SELECT COUNT(*) FROM table

The reason why this is slow is related to the MVCC implementation in PostgreSQL. The fact that multiple transactions can see different states of the data means that there can be no straightforward way for "COUNT(*)" to summarize data across the whole table; PostgreSQL must walk through all rows, in some sense. This normally results in a sequential scan reading information about every row in the table.
Some DBMSes provide the ability for "COUNT(*)" queries to work via consulting an index. Unfortunately, in PostgreSQL, this strategy does not work, as MVCC visibility information is not stored at the index level. It is necessary to actually examine the rows themselves to determine if they are visible to the transaction or not.

In MySQL, MyISAM tables cache the row count information, making this type of count operation almost instant. That is the reason why there exists so much MySQL code that uses this construct assuming it's a trivial operation. But if you're using InnoDB instead, this is no longer the case. See COUNT(*) for Innodb Tables and COUNT(*) vs COUNT(col) for notes on the limitations of MySQL in this area. MySQL designs that may be deployed on InnoDB can't assume that a full row count will be fast, and therefore are hampered by similar limitations to those present in PostgreSQL. However, InnoDB's MVCC information is present in its indexes, so an index can be used to satisfy COUNT(*) queries, even when there is no WHERE clause; a full table scan is not necessary.

It is worth observing that it is only this precise form of aggregate that must be so pessimistic; if augmented with a "WHERE" clause like

SELECT COUNT(*) FROM table WHERE status = 'something'

PostgreSQL, MySQL, and most other database implementations will take advantage of available indexes against the restricted field(s) to limit how many records must be counted, which can greatly accelerate such queries. PostgreSQL will still need to read the resulting rows to verify that they exist; MySQL may or may not, depending on the storage engine and the transaction isolation level. InnoDB generally does not need to read the rows, and can satisfy the operation from the index alone.

One popular approach for applications that need a row count but can tolerate it not including transactions that are in the middle of being committed is to use a trigger-based mechanism to count the rows in the table. In PostgreSQL, another alternative when only an approximate count is needed is to use the reltuples field from the pg_class catalog table.

Edited 2 time(s). Last edit at 04/04/2010 04:08AM by HP.