SQL Server 2005 - Rowsize effect on query performance?

Asked 16 years, 2 months ago Modified 15 years, 10 months ago

Viewed 3k times

4

Im trying to squeeze some extra performance from searching through a table with many rows. My current reasoning is that if I can throw away some of the seldom used member from the searched table thereby reducing rowsize the amount of pagesplits and hence IO should drop giving a benefit when data start to spill from memory.

Any good resource detailing such effects? Any experiences?

Thanks.

sql-server performance

Share

Improve this question

Follow

asked Oct 7, 2008 at 7:45

Torbjörn Gyllebring

18.2k ● 2 ● 31 ● 22

7 Answers

Sorted by: Highest score (default)

3

Tuning the size of a row is only a major issue if the RDBMS is performing a full table scan of the row, if your query can select the rows using only indexes then the row size is less important (unless you are returning a very large number of rows where the IO of returning the actual result is significant).

If you are doing a full table scan or partial scans of large numbers of rows because you have predicates that are not using indexes then rowsize can be a major factor. One example I remember, On a table of the order of 100,000,000 rows splitting the largish 'data' columns into a different table from the columns used for querying resulted in an order of magnitude performance improvement on some queries.

I would only expect this to be a major factor in a relatively small number of situations.

Share Improve this answer

Follow

answered Oct 7, 2008 at 8:06

mmaibaum

2,499 ● 1 ● 14 ● 11

I think you might be underestimating how many database queries are using table scans or a 'bookmark lookup' of very wide rows! I'm going to try to test this on queries I know are using entire table rows for searching. – Kenny Evitt Aug 13, 2011 at 0:00

2

I don't now what else you tried to increase performance, this seems like grasping at straws to me. That doesn't mean that it isn't a valid approach. From my experience the benefit can be significant. It's just that it's usually dwarfed by other kinds of optimization.

However, what you are looking for are iostatistics. There are several methods to gather them. A quite good introduction can be found ->here.

Share Improve this answer

Follow

answered Oct 7, 2008 at 8:05

TToni

9,375 ● 1 ● 32 ● 43

1

The sql server query plan optimizer is a very complex algorithm and decision what index to use or what type of scan depends on many factors like query output columns, indexes available, statistics available, statistic distribution of you data values in the columns, row count, and row size.

So the only valid answer to your question is: It depends :)

Give some more information like what kind of optimization you have already done, what does the query plan looks like, etc.

Of cause, when sql server decides to do a table scna (clustered index scan if available), you can reduce io-performance by downsize row size. But in that case you

would increase performance dramatically by creating a adequate index (which is a defacto a separate table with smaller row size).

Share Improve this answer

Follow

answered Oct 7, 2008 at 8:19

Jan

16k ● 5 ● 37 ● 59

1

If the application is transactional then look at the indexes in use on the table. Table partitioning is unlikely to be much help in this situation.

If you have something like a data warehouse and are doing aggregate queries over a lot of data then you might get some mileage from partitioning.

If you are doing a join between two large tables that are not in a 1:M relationship the query optimiser may have to resolve the predicates on each table separately and then combine relatively large intermediate result sets or run a slow operator like nested loops matching one side of the join. In this case you may get a benefit from a trigger-maintained denormalised table to do the searches. I've seen good results obtained from denormalised search tables for complex screens on a couple of large applications.

Share Improve this answer

Follow

answered Oct 7, 2008 at 8:30

ConcernedOfTunbridge Wells

66.5k ● 15 ● 148 ● 198

1

If you're interested in minimizing IO in reading data you need to check if indexes are covering the query or not. To minimize IO you should select column that are included in the index or indexes that cover all columns used in the query, this way the optimizer will read data from indexes and will never read data from actual table rows.

If you're looking into this kind of details maybe you should consider upgrading HW, changing controllers or adding more disk to have more disk spindle available for the query processor and so allowing SQL to read more data at the same time

SQL Server disk I/O is frequently the cause of bottlenecks in most systems. The I/O subsystem includes disks, disk controller cards, and the system bus. If disk I/O is consistently high, consider:

Move some database files to an additional disk or server.
Use a faster disk drive or a redundant array of inexpensive disks (RAID) device.
Add additional disks to a RAID array, if one already is being used.
Tune your application or database to reduce disk access operations.
Consider index coverage, better indexes, and/or normalization.

Microsoft SQL Server uses Microsoft Windows I/O calls to perform disk reads and writes. SQL Server manages when and how disk I/O is performed, but the Windows operating system performs the underlying I/O operations. Applications and systems that are I/O-bound may keep the disk constantly active.

Different disk controllers and drivers use different amounts of CPU time to perform disk I/O. Efficient controllers and drivers use less time, leaving more

processing time available for user applications and increasing overall throughput.

Share Improve this answer Follow

answered Oct 7, 2008 at 12:06

massimogentilini

4,162 ● 5 ● 31 ● 32

1

First thing I would do is ensure that your indexes have been rebuilt; if you are dealing with huge amount of data and an index rebuild is not possible (if SQL server 2005 onwards you can perform online rebuilds without locking everyone out), then ensure that your statistics are up to date (more on this later).

If your database contains representative data, then you can perform a simple measurement of the number of reads (logical and physical) that your query is using by doing the following:

```
SET STATISTICS IO ON
GO

-- Execute your query here

SET STATISTICS IO OFF
GO
```

On a well setup database server, there should be little or no physical reads (high physical reads often indicates that your server needs more RAM). How many logical

reads are you doing? If this number is high, then you will need to look at creating indexes. The next step is to run the query and turn on the estimated execution plan, then rerun (clearing the cache first) displaying the actual execution plan. If these differ, then your statistics are out of date.

Share Improve this answer

Follow

answered Oct 7, 2008 at 12:26

Mitch Wheat

300k ● 44 ● 477 ● 550

0

I think you're going to be farther ahead using standard optimization techniques first -- check your execution plan, profiler trace, etc. and see whether you need to adjust your indexes, create statistics etc. -- before looking at the physical structure of your table.

Share Improve this answer

Follow

answered Jan 30, 2009 at 15:55

mwigdahl

16.6k ● 7 ● 51 ● 64