

Re: FTS Performance in SQL 2005

Source:

<http://www.tech-archive.net/Archive/SQL-Server/microsoft.public.sqlserver.fulltext/2006-08/msg00019.html>

- *From:* "KaMa" <kaufmannma@xxxxxxxxxx>
 - *Date:* 28 Jul 2006 12:33:08 -0700
-

Hello Simon,

I'm not able to produce a STATISTICS IO on a direct query to CONTAINSTABLE (select [key] from containstable) – in this case nothing will be outputted...

Statistics IO:

```
SELECT id FROM fulltexttable WHERE CONTAINS(srchField, 'cau*' )
Table 'fulltexttable'. Scan count 2845, logical reads 8540, physical
reads 311, read-ahead reads 0, lob logical reads 0, lob physical reads
0, lob read-ahead reads 0.
```

Showplan:

```
SELECT id FROM fulltexttable WHERE CONTAINS(srchField, 'cau*' )
|--Nested Loops(Inner Join, OUTER REFERENCES:([Full-text Search
Engine].[KEY]))
|--Remote Scan(OBJECT:(CONTAINS))
|--Index
Seek(OBJECT:([DBname].[dbo].[fulltexttable].[IX_fulltexttable_3]),
SEEK:([DBname].[dbo].[fulltexttable].[id]=[Full-text Search
Engine].[KEY]) ORDERED FORWARD)
```

Now if I do a query directly to the field I would theoretically need:

Statistics IO:

```
SELECT srcID FROM fulltexttable WHERE CONTAINS(srchField, 'dan*' )
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0,
read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob
read-ahead reads 0.
Table 'fulltexttable'. Scan count 54710, logical reads 167981, physical
reads 2366, read-ahead reads 0, lob logical reads 0, lob physical reads
0, lob read-ahead reads 0.
```

Showplan:

```
--Hash Match(Aggregate,
HASH:([FT_Data].[dbo].[tFT_SearchV3].[idFilm]))
|--Nested Loops(Inner Join, OUTER REFERENCES:([Full-text Search
Engine].[KEY]))
|--Remote Scan(OBJECT:(CONTAINS))
|--Index
```

Re: FTS Performance in SQL 2005

```
Seek(OBJECT:([DBname].[dbo].[fulltexttable].[IX_fulltexttable_3]),  
SEEK:([DBname].[dbo].[fulltexttable].[idEntry]=[Full-text Search  
Engine].[KEY]) ORDERED FORWARD)
```

The index accesses (IX_fulltexttable_3) contains ID, srcID unique.

---- Another Question ----

Do you know if the Full-text/SQL Architecture has changed in the way FT uses RAM?

In SQL 2000 I had to limit SQL to take 3.5 GB (min/max) so the FT-Service will take the 0.5 GB for work.

But in SQL 2005 I was not able to reproduce this behavior (set 3.5 GB min/max but FT does not take any memory...)

Simon Sabin schrieb:

Hello KaMa,

The maximum equates to process ~4.5GB/s thats a lot.

Can you post you query plans and the output of statistics IO

Simon Sabin
SQL Server MVP
<http://sqlblogcasts.com/blogs/simons>

Hello Simon,

The PageLookups/sec are avg. 15000-17000 when doing a large full-text query, but currently I also have other processes quering the FT, so this could.

Minimum: 0

Maximum: 560613.508

Average: 53890.00

Values from ~ 100 seconds...

Do you know if the memory FT/SQL uses was merged?

Because i've set the MAX sql-server memory to 3.5 GB instead of 4.0 GB

Simon Sabin schrieb:

Hello KaMa,

its the pipe between the CPU and Memory that could be the bottleneck.

You

try drinking a gallon of water with a straw, the bootleneck is the

Re: FTS Performance in SQL 2005

straw
it can only handle a certain flow of liquid, its the pipe thats
the
bottleneck
not you (as the CPU). Now try with a 1 inch pipe the
bottleneck is
now your
ability to drink all the fluid that you can suck, the pipe is now
not
the
bottleneck instead you are.
Have a look at the Pagelookups/sec in the Buffer Manager.

I'm not saying this is the bottleneck but it is likely.

Simon Sabin
SQL Server MVP
<http://sqlblogcasts.com/blogs/simons>

Hi Simon,
But why is it that I (almost) don't see any
change in Memory or CPU
usage?
I mean if the software is running into a
bottleneck, I should see it
–
right?
Simon Sabin schrieb:

Hello KaMa,

On the memory issue,
imagine this. You need to
read 10 books and
your
in
a library. The first time you
want the book you have to
go find it
off the
shelves.
The second time you have
the book on your desk. Now
imagine the
process of
reading the data, you can
only read the books at a
certain speed.
Even though
you don't have to go hunting
for the book on the shelves

Re: FTS Performance in SQL 2005

you still
need to
read each page, if the book
is big it is going to take
longer.
In this situation the library
shelves are the disks, your
desk is
the
memory,
and you are the CPU. You
have a finite limit for
reading pages this
is the
memory bandwidth between
the CPU and the memory.
With Full text
their
is
an additional bit and that's
the communication with full
text
engine,
you
can almost think of this as
someone else reading the
books to and
writing
the page numbers you want
on a bit of paper and then
you having to
read those
pages.

So simply put you have to
reduce the amount of data
you read to get
performance

Simon Sabin

SQL Server MVP

<http://sqlblogcasts.com/blogs/simons>

Full-Text
Table:
id - int -
primary key
- clustered
(FULL
TEXT
KEY)
srcID -
varchar(30)

Re: FTS Performance in SQL 2005

(LINKING
KEY TO
ARTICLE)
srchField –
varchar(150)
–
FULL-TEXT
INDEXED
src –
varchar(15)
(SOURCE
OF
KEYWORD)
updateStamp
– timestamp
(LOCAL
TIMESTAMP)
Indexes
(fields /
description)
0 – [id] –
Primary
Key –
unique /
clustered
1 – [id],
[srcID] –
unique /
nonclustered
2 –
[srchField]
–
non-unique
/
nonclustered
3 – [srcID]
– unique /
nonclustered
4 – [id] –
non-unique
/
nonclustered
Index in use
when using
a
CONTAINS
or
CONTAINSTABLE
(1).
I can't
imagine that

Re: FTS Performance in SQL 2005

the memory
is a
bottleneck,
because I do
not
see
any change
in memory
usage when
I use a
fulltext
query...
Do you
think i
should
reduce the
max
memory of
sql server to
<3.5
GB
(currently
set to 3.5
GB of 4
GB)?
Another
question –
what means
DDL? :)
Thanks!
Simon
Sabin
schrieb:

The
artilce
title
is
won't
help
eiother
as
the
full
text
engine
doesn't
return
this.

Re: FTS Performance in SQL 2005

The
reason
that
the
CPU
usage
is
like
that
is
probably
because
you
bottleneck
is
memory.

FTS
returns
all
the
keys
that
match
the
criteria
it
will
then
join
to
your
main
table
for
each
key
returned,
if
the
indexing
is
not
right
then
you
will
end
up
reading
all

Re: FTS Performance in SQL 2005

the
data
from
the
main
table.
Whilst
this
may
be
cached
it
still
has
to
be
processed,
if
its
not
cached
it
will
be
being
read
from
disk
which
is
worse.

Your
only
option
is
to
ensure
that
you
are
only
returning
the
records
you
need
from
the
FT
engine.

Re: FTS Performance in SQL 2005

i.e
all
criteria
is
passed.
and
that
you
have
a
covering
index
on
the
columns
in
the
query.

Can
you
please
post
the
DDL
for
the
table
and
the
indexes
on
the
table.

Simon
Sabin
SQL
Server
MVP
<http://sqlblogcasts.com/blogs/simons>

Hello
Dan,
I
know
that
the
upper
queries
are

Re: FTS Performance in SQL 2005

mostly
extremes,
but
I
don't
want
do
construct
a
code
wich
can
run
into
timeouts
or
searches
which
take
~
25-40
seconds
-
even
in
the
hardest
cases...
The
problem
with
partitioning
is
that
I
need
to
give
different
ways
of
ordering
top
returned
articles
(e.g.
views,
date..)
and
so
i

Re: FTS Performance in SQL 2005

cannot
do
several
full-text
catalogs
with
access
different
tables...
The
problem
with
the
ranking-order
is
that
none
of
my
order-clauses
use
the
searched
fields
itself
(except
ordering
for
the
article
title),
and
i've
tried
CONTAINSTABLE
with
TOP_N_RANK
with
some
tests
and
got
completely
wrong
results...
No
partial
search
-
here
the

Re: FTS Performance in SQL 2005

problem
is
that
I
have
some
words
which
occur
several
thousand
times,
the
fulltexttable
is
constructed
like
a
keyword
list
–
so
if
someone
searched
for
a
keyword
that
occurs
several
times
–
the
problem
would
occur
again...
I've
already
implemented
a
caching
scenario
to
the
whole
thing
which
caches
statistics,

Re: FTS Performance in SQL 2005

counts
and
results
of
the
search...

Do
you
by
any
chance
know
how
to
give
the
SQL-FT
service
more
priority?
Because
what
really
buggers
me
is
that
if
I
execute
a
big
query
on
FTS,
almost
no
CPU/Memory
is
used
by
the
service
at
all
-
I
didn't
find
any

Re: FTS Performance in SQL 2005

documentation
on
the
memory
behaviour
of
SQL2005/FT
compared
to
SQL2000/FT
–
because
in
SQL2000/FT
you
had
to
fix
the
maximum
of
ram
SQL
uses
to
give
FT
more
memory...

I've
already
tried
to
create
a
multiple-column
scenario,
but
this
didn't
work
out
due
to
following
problem...

id
|
srcID

Re: FTS Performance in SQL 2005

|
title
|
alternate_title
|
keywords
|
names

Now
if
i've
a
field
'names'
containing
-
let's
say
-
following
data:
KaMa,
Daniel
Chrichton,
Luis
Mortan

Now
if
I
search
for
'Daniel
Mortan'
-
the
record
would
appear
even
if
the
data
is
not
correct...
This
way
i
could

Re: FTS Performance in SQL 2005

reduce
the
number
of
records
returned,
but
still
the
problem
persists
that
this
wouldn't
be
scalable,
because
if
some
keywords
would
appear
in
a
lot
of
records
–
there
still
are
a
lot
of
records
to
return
:).
Daniel
Crichton
schrieb:

KaMa
wrote
on
18
Jul
2006
01:46:46
–0700:

Re: FTS Performance in SQL 2005

Hi
Dan
–
Thanks
for
answering!

I've
got
to
pulling
the
KEY
directly
from
the
fulltexttable
because
in
cases
of
large
results
this
seems
to
be
faster.
Currently
I
have
a
solution
(still
to
slow)
wich
does
the
following
(shortened
version):
INSERT
INTO
tmpTable(uid,
id)
SELECT
@uid,
[KEY]
FROM
CONTAINSTABLE(fulltexttable,src

```
then...
INSERT
INTO
tmpTable2(uid,
srcID)
SELECT
@uid,
srcID
FROM
fulltexttable
AS
tbl
INNER
JOIN
tmpTable
AS
res
ON
ft.id=res.ID
GROUP
BY
srcID
I'm
aware
that
this
is
a
step
too
much,
but
on
large
recordsets
it
seems
to
be
faster.
```

Seems
strange
if
it
is,
as
you're
doing
pretty
much

Re: FTS Performance in SQL 2005

what
a
straight
CONTAINS
query
and
retrieving
srcID
should
be
doing.

On
your
database,
if
you
search
for
large
resultsets
(e.g.
an*
or
similar),
does
it
also
take
that
long
until
the
returning
of
all
recordsets
is
completed?

My
table
is
only
510,000
rows
so
it's
not

Re: FTS Performance in SQL 2005

as
large
as
yours.

Wouldn't
it
be
faster
to
somehow
directly
query
the
full-textservice?

When
retrieving
just
the
[KEY],
that's
what
you're
doing.
I
don't
know
of
a
way
to
access
the
FTS
service
outside
of
SQL
Server,
if
there
is
a
way
at
all.

Re: FTS Performance in SQL 2005

I've
already
had
an
index
containing
ID,
srcID
and
src
–
wich
was
accessed
with
CONTAINS
–
so
that
shouldn't
be
a
problem...

To
compare
the
queries,
put
them
all
into
Query
Analyser
and
turn
on
Show
Execution
Plan,
and
then
run
the
whole
lot
–
the
Execution
Plan

window
will
then
give
you
a
percentage
query
cost
relative
to
the
whole
batch,
making
it
easier
to
spot
which
appears
to
be
the
best
solution.
Basing
it
on
times
is
a
bad
idea
as
caching
can
affect
the
results.

If
you
can
do
so,
try
to
build
your
queries

to
match
on
whole
words,
or
reduce
the
number
of
results
you
pull
back.
The
more
results
you
have,
the
worse
your
performance
will
be
as
SQL
Server
will
have
to
look
up
additional
data,
and
with
a
very
large
number
of
rows
you
end
up
with
a
Merge
Join/Inner
Join

which
adds
overhead
on
the
Sort.
For
example,
if
I
run
the
following,
the
query
cost
for
each
is:

```
select
ID
from
STK
where
CONTAINS(STK.QuickSearch,
"windows")
Results:
1272,
Cost:
14.8%
(nested
loop)
```

```
select
ID
from
STK
where
CONTAINS(STK.QuickSearch,
"window*")
Results:
1450,
Cost:
14.9%
(nested
loop)
```

```
select
ID
from
```

Re: FTS Performance in SQL 2005

STK
where
CONTAINS(STK.QuickSearch,
"win*")
Results:
7269,
Cost:
15.28%
(merge
join
with
sort)

select
ID
from
STK
where
CONTAINS(STK.QuickSearch,
"wi*")
Results:
60325,
Cost:
23.78%
(merge
join
with
sort)

select
ID
from
STK
where
CONTAINS(STK.QuickSearch,
"w*")
Results:
176914,
Cost:
31.16%
(hash
match)

As
you
can
see,
the
more
rows
that

are
returned
the
more
work
is
done
to
pull
information
from
the
table.

Comparing
on
the
FTS
search
itself,
it's
obvious
that
returning
a
lot
of
rows
is
a
bad
idea:

```
select  
[KEY]  
from  
CONTAINSTABLE(STK,QuickSearch,  
"windows")  
Cost:  
3.09%
```

```
select  
[KEY]  
from  
CONTAINSTABLE(STK,QuickSearch,  
"window*")  
Cost:  
3.12%
```

```
select  
[KEY]
```

from
CONTAINSTABLE(STK,QuickSearch,
"win*")
Cost:
4.15%

select
[KEY]
from
CONTAINSTABLE(STK,QuickSearch,
"wi*")
Cost:
30.02%

select
[KEY]
from
CONTAINSTABLE(STK,QuickSearch,
"w*")
Cost:
59.62%

Notice
how
the
last
two
cost
significantly
more
due
to
the
number
of
rows
being
returned.
Retrieving
data
from
the
FTS
engine
is
a
comparitively
slow
process.
This
is

much
more
obvious
when
you
use
TOP_N_BY_RANK
to
reduce
the
number
of
results.

```
select  
[KEY]  
from  
CONTAINSTABLE(STK,QuickSearch,  
"w*")  
Cost:  
97.95%
```

```
select  
[KEY]  
from  
CONTAINSTABLE(STK,QuickSearch,  
"w*",1000)  
Cost:  
2.05%
```

If
you
can
find
a
way
to
work
with
the
TOP_N_BY_RANK
and
CONTAINSTABLE
you
should
be
able
to
optimise
your
queries

significantly.
Luckily
I
rarely
have
queries
that
take
longer
than
2
or
3
seconds
to
run
due
to
only
having
510,000
rows,
and
in
most
cases
less
than
2,000
results
for
any
given
search.
I
notice
in
your
original
post
that
you
say
that
you
cannot
do
this
due
to
the

results
being
"ordered
totally
wrong".
There
is
nothing
stopping
you
using
an
ORDER
BY
on
the
returned
results
to
change
the
sort
order,
and
using
the
ISABOUT,
NEAR,
and
other
keywords
in
FTS
to
adjust
the
way
the
rank
is
calculated
to
try
to
ensure
that
you
get
just
the
results

Re: FTS Performance in SQL 2005

that
you
need
from
the
FTS
search.

Dan