

Re: Possible memory leak using \$sth->{NAME} ?

Source: <http://coding.derkeiler.com/Archive/Perl/perl.dbi.users/2006-06/msg00034.html>

- *From:* david.brewer@xxxxxxxxxx (David Brewer)
 - *Date:* Tue, 6 Jun 2006 16:11:45 -0700
-

I have not yet been able to replicate in mysql but I spent some time paring down the query to the minimum version that causes the memory leak. The results are interesting and I think may point toward DBD::ODBC being the issue. I don't understand the internals well enough to be sure of that, though.

The basic idea behind the original query was to do a complicated search, store the resulting distinct ids into a table variable called @search_hits, and then use that table joined to some other tables to select the final results. I cut out everything that I could remove while still seeing the leak. Here is the simplest query I could come up with that exhibits the behavior:

```
#####  
DECLARE @search_hits TABLE (  
ObjectID int DEFAULT(0)  
);  
  
INSERT INTO @search_hits (ObjectID) VALUES (1);  
  
SELECT TOP 0  
*  
FROM  
Objects;  
#####
```

Basically, it has to have the following properties to exhibit the large leak:

- 1) You must create a table variable.
- 2) You must insert at least one row into the table variable.
- 3) You must select from a table that contains "text" columns. The more columns you select, especially those that contain long text, the greater the leak. Note that you don't have to select ANY rows in order to cause the leak!
- 4) After executing the query, you must do something which triggers the lookup of column information. I've found that either \$sth->{NAME} or \$sth->fetchrow_hashref() will cause the behavior.

If any of these conditions is not met in the query, the leak does not

Re: Possible memory leak using \$sth->{NAME} ?

happen. I'm guessing this means it will not be reproducible in mysql, as if I recall correctly mysql doesn't have an equivalent to a table variable.

Do DBI drivers typically store information about a query aside from information about the actual result of the query? My original assumption was that if you did a complicated query, it was only the results of the query that would make a difference to the memory usage of perl. It surprised me to find that creating a table variable, doing an insertion, and then ignoring that table could have an effect on memory usage in the perl process.

I've included the entire script for reproducing the problem below, this time including the SQL. Thanks again!

```
#####
```

```
use strict;
use warnings;
use DBI;

my $dsn = qq{DBI:ODBC:driver={SQL
Server};Server=SERVERNAME;database=DBNAME;uid=DBUSER;pwd=PASSWORD;};

# 'Objects' must contain several columns of type 'text' to get
# a substantial leak
my $sql = q{
DECLARE @search_hits TABLE (
ObjectID int DEFAULT(0)
);

INSERT INTO @search_hits (ObjectID) VALUES (1);

SELECT TOP 0
*
FROM
Objects;
};

for my $i (1..50) {
print "Executing iteration $i... \n";
my $dbh = DBI->connect($dsn);
$dbh->{LongReadLen} = 20000;
my $sth = $dbh->prepare($sql);
$sth->execute();
$sth->{NAME};
$sth->finish;
$dbh->disconnect();
undef($dbh);
}
```

Re: Possible memory leak using \$sth->{NAME} ?

#####

On 6/6/06, David Brewer <david.brewer@xxxxxxxx> wrote:

OK, I will attempt to replicate in mysql and get back to you. In the meantime, I have run my test script in some older versions of DBI to see if that helps anything. I tried 1.49, 1.48, 1.47, 1.45, 1.41, and 1.38. I saw pretty the same behavior in each version.

Thanks,

David

On 6/6/06, Tim Bunce <Tim.Bunce@xxxxxxxx> wrote:

> Can you get DBD::mysql to leak in the same way (\$sth->{NAME})?

> I need to be able to duplicate the problem and I can't do that

> easily with ODBC or ADO.

>

> Tim.

>

> On Tue, Jun 06, 2006 at 12:14:36PM -0700, David Brewer wrote:

>> I've tried this with a couple of different drivers now -- DBD::ADO and

>> DBD::mysql. In both cases the same simple connect and disconnect

>> routine seems to leak 1 SV per execution.

>>

>> The ADO example is exactly the same as the previous example, except I

>> substituted "DBI:ADO" for "DBI:ODBC". The MySQL example is a little

>> different and is included below as 'Example 1'.

>>

>> I'm actually not too concerned about that particular leak -- it seems

>> to be a fairly insignificant amount of memory. It's just the simplest

>> example of a leak I could construct and I was hoping that it might be

>> a symptom of some mistake that I might be making. The problem that

>> led me to it is a much larger leak that occurs when try to get the

>> column names from a very complicated query (either directly using

>> \$sth->{NAME} or indirectly by calling \$sth->fetchrow_hashref).

>>

>> In the case of one query I am using it seems to be leaking about 144k

>> of memory per trip through the loop. The same loop leaks about 27.5k

>> each time if I use the default 'LongReadLen', so it is somehow related

>> to column size. I'm not fetching any data after executing the query,

>> just touching \$sth->{NAME} to force it to find the column names. You

>> can see an example of this loop as Example 2, below.

>>

>> If I comment the line with \$sth->{NAME}, then the leak becomes so

>> small as to be negligible.

>>

>> The way I am computing the size of the leak in this case is running a

>> version of the script where the loop executes once, then looking at

>> the memory taken up by the perl process. Then I execute a version of

>> the script where the loop executes N times, subtract the memory taken

Re: Possible memory leak using \$sth->{NAME} ?

```
>> in the first test from the memory taken by the second test, and divide
>> by N-1 to get the average memory leaked per pass.
>>
>> I have also done the same test using Apache::Leak to measure SVs being
>> leaked, and there are apparently 4 SVs leaked per pass. They just
>> happen to be relatively large ones, I guess. :-)
>>
>> I am very willing to run any other tests you might suggest. I am
>> thoroughly mystified at this point and eager to get to the bottom of
>> this.
>>
>> #####
>> # Example 1:
>> #####
>> use strict;
>> use warnings;
>> use DBI;
>>
>> my $dsn = "DBI:mysql:database=DBNAME;host=DBHOST;port=3306";
>> my $user = "USER";
>> my $password = "PASSWORD";
>>
>> use Apache::Leak;
>> leak_test {
>> for (1..50) {
>> my $dbd = DBI->connect($dsn, $user, $password);
>> $dbd->disconnect();
>> undef($dbd);
>> }
>> };
>> #####
>> #####
>> # Example 2
>> #####
>>
>> use strict;
>> use warnings;
>> use DBI;
>>
>> my $dsn = qq{DBI:ODBC:driver={SQL
>> Server};Server=SERVERNAME;database=DBNAME;uid=DBUSER;pwd=PASSWORD;};
>> my $options = { RaiseError => 1 };
>>
>> # in reality this is a enormous query involving table variables
>> # in MSSQL -- excluded here for simplicity. I can include
>> # it if you want to see it.
>> my $sql = ";
>>
>> for my $i (1..50) {
```

Re: Possible memory leak using \$sth->{NAME} ?

```
>> print "Executing iteration $i... \n";
>> my $dbd = DBI->connect($dsn, $options);
>> $dbd->{LongReadLen} = 20000;
>> my $sth = $dbd->prepare($sql);
>> $sth->execute();
>> $sth->{NAME};
>> $sth->finish;
>> $dbd->disconnect();
>> undef($dbd);
>> sleep(1);
>> }
>>
>>
>> #####
>>
>> On 6/6/06, David Brewer <david.brewer@xxxxxxxx> wrote:
>>> Sure, I'd be glad to check into that.
>>>
>>> I think Apache::Leak already runs the code twice -- first it runs the
>>> code to make sure that anything that would get cached is already in
>>> memory. Then it measures the memory usage, runs the code again, and
>>> measures the memory usage a final time to determine how much was
>>> leaked.
>>>
>>> I made a new script which you can see below. This one does a similar
>>> leak test but repeats it 50 times. leak_test reports 50 SV leaked, so
>>> it seems like it's consistent.
>>>
>>> I will start trying some different drivers and report back my results
>>> shortly. Thanks for your response!
>>>
>>> David
>>>
>>> #####
>>>
>>> use strict;
>>> use warnings;
>>> use DBI;
>>>
>>> my $dsn = qq{DBI:ODBC:driver={SQL
>>> Server};Server=SERVERNAME;database=DBNAME;uid=DBUSER;pwd=PASSWORD;};
>>> my $options = { RaiseError => 1 };
>>>
>>> use Apache::Leak;
>>> leak_test {
>>> for (1..50) {
>>> my $dbd = DBI->connect($dsn, $options);
>>> $dbd->disconnect();
>>> undef($dbd);
```

Re: Possible memory leak using \$sth->{NAME} ?

```
>>> }
>>>};
>>>
>>>#####
>>>
>>>On 6/6/06, Tim Bunce <Tim.Bunce@xxxxxxxx> wrote:
>>>> On Mon, Jun 05, 2006 at 07:51:22PM -0700, David Brewer wrote:
>>>>> OK, I have pared my problem down to a small test script and I've cut
>>>>> away a lot of the modules I'm using that don't seem to be part of the
>>>>> issue. The test script is included at the end of this message.
>>>>
>>>> Thanks.
>>>>
>>>>> This small script doesn't leak much memory, but it's surprising to me
>>>>> that it leaks at all. Essentially, I just connect to a database and
>>>>> then disconnect from it, and Apache::Leak reports that this process
>>>>> leaks 1 SV. If I add a simple query then Apache::Leak reports I leak
>>>>> 4 SVs.
>>>>
>>>>> 'leaks' from one-off calls are rarely real leaks, they're often just
>>>>> internal caching of one kind or another.
>>>>
>>>>> Real leaks leak in proportion to the number of calls made. I'd expect
>>>>> you to be able to say something like "each call to foo leaks N scalars"
>>>>> (because 100 calls leak X and 101 calls leak X+N).
>>>>
>>>>> Can you check for that? And can you also try a different driver or two?
>>>>
>>>>> Tim.
>>>>
>>>>>> I am using DBI 1.50 with DBD::ODBC 1.13. This is on a Windows XP
>>>>>> machine, using ActivePerl 5.8.8 (I was using 5.8.7 previously with the
>>>>>> same issue). I am talking to a MSSQL Server 2000 database.
>>>>>
>>>>>> #####
>>>>>> use strict;
>>>>>> use warnings;
>>>>>> use DBI;
>>>>>>
>>>>>> my $dsn = qq{DBI:ODBC:driver={SQL
>>>>>> Server};Server=SERVERNAME;database=DBNAME;uid=DBUSER;pwd=PASSWORD;};
>>>>>> my $options = { RaiseError => 1 };
>>>>>>
>>>>>> use Apache::Leak;
>>>>>> leak_test {
>>>>>> my $dbd = DBI->connect($dsn, $options);
>>>>>> $dbd->disconnect();
>>>>>> };
>>>>>> #####
>>>>>>
```

Re: Possible memory leak using \$sth->{NAME} ?

```
>>>> On 6/2/06, David Brewer <david.brewer@xxxxxxxx> wrote:
>>>>> I am having what appears to a memory leak problem on a mod_perl
>>>>> project I am working on. On the worst case web page (a search results
>>>>> page) I am leaking an average of about 160k per page load! I think
>>>>> I've finally isolated the problem and it appears to be related to DBI.
>>>>> It's very possible that I am doing something wrong to cause the
>>>>> problem. :-)
>>>>>
>>>>> First of all, some version and module information. I am using DBI
>>>>> 1.50 with DBD::ODBC 1.13. I am using persistent database connections
>>>>> via the Apache::DBI module. The project is using mod_perl and
>>>>> Apache::ASP.
>>>>>
>>>>> I isolated the problem by commenting out great swaths of code until
>>>>> the problem went away, and then slowly adding them back in until it
>>>>> reappeared. My first thought was that it had something to do with
>>>>> fetchrow_hashref. I have a loop like this:
>>>>>
>>>>> while ($row = $sth->fetchrow_hashref) {
>>>>> # do stuff here
>>>>>}
>>>>>
>>>>> All of the functionality inside the loop has been commented, but my
>>>>> memory leak still happens. However, if I comment the loop entirely,
>>>>> the leak goes away (well, about 158k of it at least!).
>>>>>
>>>>> If I replace the loop with something like:
>>>>>
>>>>> while ($row = $sth->fetchrow_arrayref) {
>>>>> # do stuff here
>>>>>}
>>>>>
>>>>> ... no leak. I need to get at some of the column names, though, so I
>>>>> added this line before the loop:
>>>>>
>>>>> my $column_names = $sth->{NAME};
>>>>>
>>>>> ... and the leak was back! It stays even if I don't save the column
>>>>> names into a variable, but just touch them:
>>>>>
>>>>> $sth->{NAME};
>>>>>
>>>>> In fact, it even stays if I remove the loop entirely and just include
>>>>> the line above! Any ideas why this might be happening and what I
>>>>> could do to fix it or work around it? Is there possibly something I'm
>>>>> doing wrong here?
>>>>>
>>>>> My next step is going to be to try to make some kind of simple test
>>>>> outside of the framework of my web project that reproduces the same
>>>>> behavior. I'll post that here when I have it.
>>>>>
```

Re: Possible memory leak using \$sth->{NAME} ?

>>>>>Thanks in advance for any insight,
>>>>>
>>>>>David Brewer
>>>>>
>>>>
>>>
>>
>>
>>
>