Subject: Re: Strange Java bridge problem !!!
Posted by Michael Wallace on Fri, 13 Jan 2006 21:21:17 GMT
View Forum Message <> Reply to Message

Well, most of the IDL-Java Bridge threads have my name in their
somewhere, so I guess my name belongs in this one as well.

It sounds very much like you have a memory leak somewhere. Typically
when hotspot crashes like how you've described, it means that it has run
out of memory. This is either due to a memory leak in the Java code or
due to your computer's memory being used up to the point that hotspot
can't do what it needs. My guess is that you have memory leak -- it's
more likely and I remember running into something very similar once upon
a time.

It is painfully easy to create memory leaks using the IDL-Java Bridge
and they can be painfully hard to find. One of the common memory leaks
results from reusing a variable name without first destroying it.

```
java_var = obj_new('IDLjavaObject', ...)  ; line 1
java_var = obj_new('IDLjavaObject', ...)  ; line 2
obj_destroy, java_var
```

Here you have a basic memory leak. On line 1, a variable is created on
the heap and is referenced by java_var. On line 2, a new variable is
created on the heap and the existing java_var reference is changed to
point to this new heap variable. When you do the obj_destroy, the java
object created on line 2 is destroyed. But the java object created on
line 1 is still in the heap. And you can't use obj_destroy because you
have no reference to the object (java_var was reassigned in line 2).

A similar problem can happen if you create a java object but don't
assign it to a variable. Because the variable name is used as a
reference, you have no way to use obj_destroy. For example:

```
my_cool_func(obj_new('IDLjavaObject', ...))
```

Here I'm passing a java object into a function. If there is no
obj_destroy within the function itself, I have no way to destroy it. A
heap variable is being created, but without any way to get to it, you
can't destroy it.

If you have a small program, you might not notice the memory leak.
However, if you iterate many times over the code with the leak,
eventually you'll run out of memory. Because the memory that is being
leaked is leaked into the Java space, it is hotspot rather than IDL that
crashes due to lack of memory.

It's pretty easy to figure out if you have a memory leak like this. You can see the objects and pointers stored on the heap with the help command (help, /heap). Try running your program with just one loop and before you cleanup your objects, take a look at the number of heap variables. Run your program again, but this time loop several times before looking at the number of heap variables. If you see that the number has really shot up, it means you're not destroying all of your objects.

Also, I would recommend that you create as few objects as possible within your loop. I don't know how your code is structured, but I know that you should only have to create one database connection and then just reuse it instead of creating a new connection on each iteration.
It might even be possible to reuse your statement and resultSet objects (and other objects if you have them) depending on the nature of the queries. My point is, the fewer objects you have, the better.

Hope this helps,
Mike

Antonio Santiago wrote:
> Hi group,
>
> as I can see in older post I am not the only one that uses JDBC to
> emulate de Dateminer.
>
> I have an strange problem (because this I need some help :)).
> Well, I am running a Linux box with IDL, MySQL and Java v1.5.
> I have create a little java class to create a JDBC connection and a
> statement object.
> From this statement (and inside IDL) make a query, extract some info
> and   close and OBJ_DESTROY de resultSet object.
>
> The strange problem is that all is right since some higer number of
> queries are made, that is:
>
> 1 iter. - Create JDBC connection, get statement object, make query,
> extract info and close and OBJ_DESTROY object.
>
> 2 iter. - The same.... ok.
> 3 iter. - The same... ok.
>
> n iter. - Java Hotspot crashed !!!! What ??? This is not an exception
> problem.
>
> I have made a little program in Java with the same behavior (connection,
> statement, query, etc). I execute it a hundred of times and only in one
> execution it returns me an exception on socket creation when try to

> stablish the connection to database.
>
> I think the problem is IDL don't handles in a right way the exception
> that is produced.
>
>
> Any help (or joke to make me happy) will be apreciate :(
>
>