Why is it bad practice to make multiple database connections in one request?

Asked 16 years, 3 months ago Modified 9 years, 8 months ago

Viewed 14k times

5

A discussion about Singletons in **PHP** has me thinking about this issue more and more. Most people instruct that you shouldn't make a bunch of DB connections in one request, and I'm just curious as to what your reasoning is. My first thought is the expense to your script of making that many requests to the DB, but then I counter myself with the question: wouldn't multiple connections make concurrent querying more efficient?

How about some answers (with evidence, folks) from some people in the know?

database resources database-connection

Share

Improve this question

Follow

edited Apr 17, 2015 at 18:10

Trikaldarshiii
11.3k ● 16 ● 69 ● 95

asked Aug 26, 2008 at 16:53

Brian Warshaw
23k ● 9 ● 54 ● 72

5 Answers

Sorted by: Highest score (default)

9

Database connections are a limited resource. Some DBs have a very low connection limit, and wasting connections is a major problem. By consuming many connections, you may be blocking others for using the database.

Additionally, throwing a ton of extra connections at the DB doesn't help anything unless there are resources on the DB server sitting idle. If you've got 8 cores and only one is being used to satisfy a query, then sure, making another connection might help. More likely, though, you are already using all the available cores. You're also likely hitting the same harddrive for every DB request, and adding additional lock contention.

If your DB has anything resembling high utilization, adding extra connections won't help. That'd be like spawning extra threads in an application with the blind hope that the extra concurrency will make processing faster. It *might* in some certain circumstances, but in other cases it'll just slow you down as you thrash the hard drive, waste time task-switching, and introduce synchronization overhead.

Share Improve this answer Follow

answered Aug 26, 2008 at 17:03

Derek Park
46.8k ● 16 ● 59 ● 76

4

It is the cost of setting up the connection, transferring the data and then tearing it down. It will eat up your performance.

Evidence is harder to come by but consider the following...

Let's say it takes x microseconds to make a connection.

Now you want to make several requests and get data back and forth. Let's say that the difference in transport time is negligable between one connection and many (just ofr the sake of argument).

Now let's say it takes y microseconds to close the connection.

Opening one connection will take x+y microseconds of overhead. Opening many will take n * (x+y). That will delay your execution.

Share Improve this answer Follow

answered Aug 26, 2008 at 16:56

Craig
11.9k ● 13 ● 45 ● 62

2

Setting up a DB connection is usually quite heavy. A lot of things are going on backstage **(DNS resolution/TCP connection/Handshake/Authentication/Actual Query)**.

I've had an issue once with some weird DNS configuration that made every TCP connection took a few seconds before going up. My login procedure (because of a complex architecture) took 3 different DB connections to complete. With that issue, it was taking forever to log-in. We then refactored the code to make it go through one connection only.

Share Improve this answer Follow

answered Aug 26, 2008 at 16:59

Vincent

22.9k ● 18 ● 59 ● 61

1

We access Informix from .NET and use multiple connections. Unless we're starting a transaction on each connection, it often is handled in the connection pool. I know that's very brand-specific, but most(?) database systems' cilent access will pool connections to the best of its ability.

As an aside, we did have a problem with connection count because of cross-database connections. Informix supports synonyms, so we synonymed the common offenders and the multiple connections were handled server-side, saving a lot in transfer time, connection creation overhead, and (the real crux of our situtation) license fees.

answered Aug 26, 2008 at 17:02

hometoast

11.8k ● 5 ● 42 ● 58

0

I would assume that it is because your requests are not being sent asynchronously, since your requests are done iteratively on the server, blocking each time, you have to pay for the overhead of creating a connection each time, when you only have to do it once...

In Flex, all web service calls are automatically called asynchronously, so you it is common to see multiple connections, or queued up requests on the same connection.

Asynchronous requests mitigate the connection cost through faster request / response time...because you cannot *easily* achieve this in PHP without out some threading, then the performance hit is greater then simply reusing the same connection.

that's my 2 cents...

answered Aug 26, 2008 at 16:58

mmattax

27.6k ● 42 ● 117 ● 151