COMP220 Performance testing notes University of Liverpool

Author Sebastian Coope

Most modern applications are multi-tiered, this means there are many components which work together to provide the users experience.

Client side
Typically a web page running in a browser (running because most web pages run code such as Javascript). It also can be an native application running on iOS, Android device or other mobile platform, there are also other possible client applications like a PC application.

Server side
Typically split into 2 components as minimum, an application server and a database server.

Server side Application Server
This component generates the content that the client displays and also receives requests from the client which it responds to. If the client is a browser this generates web content (HTML, Javascript, CSS etc). Typical options here are PHP, .NET and Java servlet services (e.g. Tomcat) but there are many other possible options.

Server side Database server
Usually a relational database server such as MySQL, Oracle, MS-SQL, Informix etc.

Performance testing client side

Typical you need to test in many different client target devices (for example across different browsers, different versions of iOS, Android and different hardware platforms). This makes sure that the user experience is still good even if they have a less powered device (for example with smaller memory and slower CPU). This cross platform testing can be done using a tool such a Selenium which allows the test to be carried out natively within the browsers environment.

The other issue with client testing is to test the client when connecting over a slow internet connection, this is to make sure that the user's experience is still good when connected over a slow network. To help testing with slow connections you can use a proxy to act as an intermediary this slows down the internet connection to the application so the user's experience can be fully tested.

Performance testing Application Server

When testing the application server you need to test it with a heavy load i.e. many user transactions per second The load is typically measured in Erlangs, the Erlang load = Time between request/Average service time of request. For a given server the service time will be the amount of CPU time needed to execute the requests code and may be difficult to measure for a given application. This is because when calculating the time for code to run, when accessing for example the database the CPU may be idle while waiting for the database to respond.

So when benchmarking the application server the database request code should be ideally stubbed out and simulated data should be used instead.

It is worth noting the following:

Given that traffic is often random in nature even if the load of a server is an average of less than one Erlang the buffers inside can overflow due to a sudden burst of requests. For this reason servers have to have large memory buffers to store incoming traffic and should ideally not be heavily loaded, e.g. < 0.5 Erlangs.

If the average load of a server is > 1 Erlang, however large its buffers are they will always eventually overflow.

Tools that can be used to stress test the website include tools such as the Low Orbit Ion Cannon and Loadster.

Performance testing Database server

Database servers are different from the application servers in that their performance is very often effected by the current state of the server. If the servers tables have a very large number of rows it will take longer to search the tables and longer to re-build indexes on insertion of data.

So the performance of an SQL database depends on a range of factors. The first is the amount of load on the database in terms of how many queries per second it is handling.

The second is the state of the database tables in terms of how many rows they contain and their current lock status. It is therefore important to test the database in a range of conditions by populating the database tables with a large number of rows. The number of rows should be increased in the database tables until queries on the tables take an unacceptable length of time. This can be seen as an upper limit on the acceptable row load for the table.

When testing database a range of queries should be used, these should be represent the typical types of queries that the application would be performing on the database.

A lot of databases (including MySQL) have a query cache, this stores the results of recent queries, if a query is repeated on a set of tables that haven't changed the query is returned from the cache. This means if a query is repeated the result may be got from cache the performance measured will be very fast and not represent the actual time that the query would take in reality. So for each query if the query is contained within the caches internal storage or state then the query will take a lot less time. If the query cache is very small most database queries will be cache misses and the query will take longer.

To get round the caching problem, either the cache can be switched off or the memory allocated to it should be very small or the query can be modified or the tables updated.

Other ways of monitoring the database can involve looking at database logs such as the slow query log. This will find out the worst case SQL queries and then they can be used as test queries optimised or the tables that the queries run on be broken up or the database scaled up in terms of its performance.

Tools that can be used to load a database include applications such as mysqlslap.