(/learn)

How to Improve and Test the Scalability of our Application?

In this lesson, we will cover how to improve and test the scalability of our application.

We'll cover the following

Here are some of the common and best strategies to fine-tune the performance of our web application. If the application is performance-optimized it can withstand more traffic load with less resource consumption as opposed to an application that is not optimized for performance.

Now you might be thinking, "why are you talking about *performance* when we should be talking about *scalability*?"

Well, the application's performance is directly proportional to scalability. If an application is not performant it will certainly not scale well. These best practices can be implemented even before the real pre-production testing is done on the application.

So, here we go.

Tuning the performance of the application – Enabling it to scale better#

Profiling#

Profile the hell out of your app. Run *application profiler* and *code profiler*. See which processes are taking too long and which are eating up too many resources. Find out the bottlenecks. Get rid of them.

Profiling is the dynamic analysis of our code. It helps us measure the space and the time complexity of our code and enables us to figure out issues like concurrency errors, memory errors and robustness and safety of the program. This Wikipedia resource contains a good list of performance analysis tools used in the industry (https://en.wikipedia.org/wiki/List_of_performance_analysis_tools)

Caching#

Cache wisely, and cache everywhere. Cache all the static content. Hit the database only when it is really required. Try to serve all the read requests from the cache. Use a write-through cache.

CDN#

Use a *Content Delivery Network (CDN)*. Using a CDN further reduces the latency of the application due to the proximity of the data from the requesting user.

Data compression#

Compress data. Use apt compression algorithms to compress data, and store data in the compressed form. As compressed data consumes less bandwidth, consequently, the download speed of the data on the client will be faster.

Avoid unnecessary client server requests#

Avoid unnecessary round trips between the client and server. Try to club multiple requests into one.

These are a few of the things we should keep in mind in the context of application performance.

Testing the scalability of our application#

Once we are done with the basic performance testing of the application, it is time for capacity planning, provisioning the right amount of hardware and computing power.

The right approach for testing the application for scalability largely depends on the design of our system. There is no definite formula for this. Testing can be performed at both the hardware and the software level. Different services and components need to be tested both individually and collectively.

During the scalability testing, different system parameters are taken into account such as the *CPU usage, network bandwidth consumption, throughput, number of requests processed within a stipulated time, latency,*

memory usage of the program, end-user experience when the system is under heavy load etc.
In this testing phase, simulated traffic is routed to the system, to study how the system behaves under the heavy load, and how the application scales under the heavy load. Contingencies are planned for unforeseen situations.

As per the anticipated traffic, the appropriate hardware and computational power are provisioned to handle the traffic smoothly with some buffer.

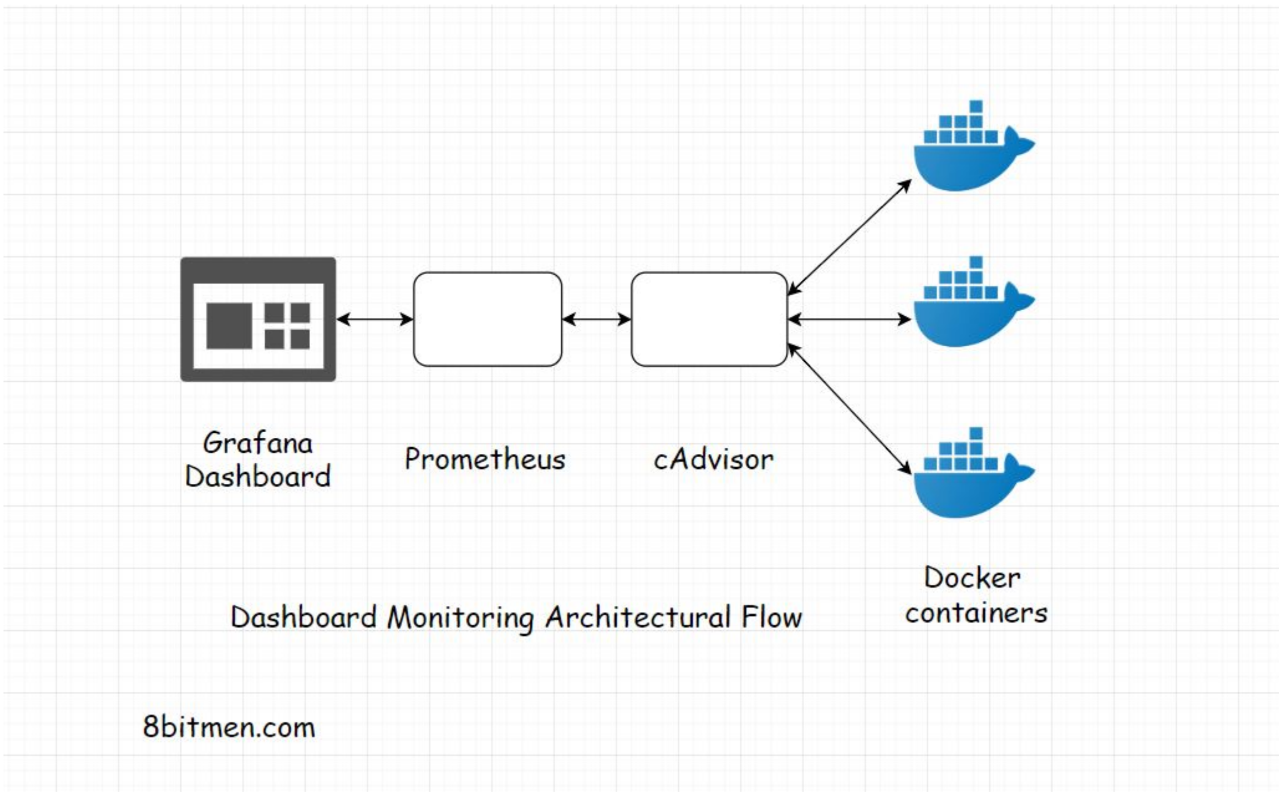
Several *load* and *stress* tests are run on the application. Tools like *JMeter* are pretty popular for running concurrent user tests on the application if you are working on a *Java* ecosystem. There are a lot of cloud-based testing tools available that help us simulate test scenarios just with a few mouse clicks.

Businesses test for scalability all the time to get their systems ready to handle a traffic surge. If it's a sportings website it would prepare itself for the sports event day. If it's an e-commerce website it would make itself ready for festival season.

Read how production engineers support global events on Facebook. (https://engineering.fb.com/production-engineering/how-production-engineers-support-global-events-on-facebook/)

Also, how Hotstar a video streaming service scaled with over 10 million concurrent users (https://www.8bitmen.com/how-hotstar-scaled-with-10-3-million-concurrent-users-an-architectural-insight/)

In the industry tech like *Cadvisor*, *Prometheus* and *Grafana* are pretty popular for tracking the system via web-based dashboards.



I’ve written an article on it in case you want to read more about the pre-production monitoring. (https://www.8bitmen.com/what-is-grafana-why-use-it-everything-you-should-know-about-it/)

← Back
Primary Bottlenecks That Hurt the Sca...

Next →
Scalability Quiz

✓ Mark as Completed

Report an Issue