Case Study

SMS Delivery Testing Service

Objective

Detect flaws and fix the Client's system which provides sms delivery testing service.

Challenge

From the business perspective, the Client's system equips end users to test sms delivery services. The Client's end users – mostly SMS aggregators and mobile operators – can schedule sms delivery tests and receive relevant statistics via their emails. There are two options of using the testing service. End users can either connect it to their systems via APIs or there's an option to log into the Client's platform and manage running tests from there.
The Client approached CPCS to debug the system, which used to crash once in a while.

Solution

CPCS's engineers started the process of debugging the code developed by a third party. At the initial stage of the process, we applied some basic code-inspection methods to learn the process logic and detect possible flaws.
According to the process logic, the system's end users add their SMPP servers' parameters and choose the other necessary ones (e.g. vendors, SMSC, country and networks, sender ID and message content, number of executions, etc.) to start using the testing service and as a result receive the data requested. This is enabled by a system module written in PHP and using Redis to store its data.

What we saw after performing a series of dynamic QA techniques, was that fork processes had definite flaws. Every time different end users were sending their requests simultaneously, the system crashed.

This output validation error wasn't that easy to debug. The most common solution – to update Redis version from v.3 to v.5 – helped only during the first week and the crash happened again.
All forked child processes used a single Redis connection object, which means they were executed via one singular connection. It is usually quite normal for similar processes, but for our particular case the idea to change that proved to be the game changer. We discovered that when two different fork processes were trying to request the data simultaneously, the Redis connection didn't understand what output should be sent to what receiver.
We fixed the whole system by our engineer's solution to build different Redis connections for different child processes.

Industry

Telecommunications, any industry

Application

System flaws detection, fixing a highload project's code written by a third party

Quick Fact

CPCS solved the Client's system crashing problem by redesigning the connection logic of microservices' child processes to Redis.

Being capable both to develop and to debug highload projects, CPCS team will be happy to learn more about your particular business needs!

About CPCS

From artificial neural networks to custom electronics design, highload and big data projects. CPCS is a team of 100+ senior-level dedicated engineers open to go through an in-depth feature-by-feature product discovery process with you. We believe you would also benefit from our client-first approach, transparency of all our magic, as well as fair and flexible rates.

hello@cpcs.team