Performance and load testing results of WebSpellChecker v5.5.9

This document describes the results of performance tests for WebSpellChecker Web API. The performance and load were tested and accessed depending on the following setup:

- Certain number of users accessing the server simultaneously (10/20/50/100 users);
- Cache enabled for spell checking purposes;
- Certain hardware and software used (EC2 m5.large instance with 2 CPU and 8 GB RAM);
- Number of words to be checked (1K words or 6K characters);
- Number of spelling and grammar problems in the text (50 grammar problems | 200 misspellings);
- Type of the language used for check (17 default languages).

Testing goal

Our main goal was to observe the **response time** of text processing and **CPU utilization** on the server in the case when 10/20/50/100 users send simultaneous requests on various languages to the server with WebSpellChecker v5.5.9.

Environment and testing tool

- We used Apache Jmeter 5.1.1 as a performance measuring tool.
- The machine used was AWS EC2 m5.large instance with 2 CPU and 8 GB RAM.
- The version of the WebSpellChecker Server package is 5.5.9 (released on April 21, 2020).

Testing process

We have run our tests continuously increasing the number of users accessing it for each of the languages in the default language group (17 languages). The cache setting was enabled for all sets of tests. The following combinations of text to be checked are used for each language and for each user tier:

- 1K words (6K characters) with 50 grammar problems, 200 misspellings;
- 1K words with 50 grammar problems only;
- 1K words with 200 misspellings only.

The measured were the response time and CPU utilization.

Observations

Our observations are presented in the charts below.

Response time and CPU utilization (only 200 misspellings)

Chart below represents **response time** results aggregated by language and user tiers when there are **only 200 misspellings** in text of 1K words size.

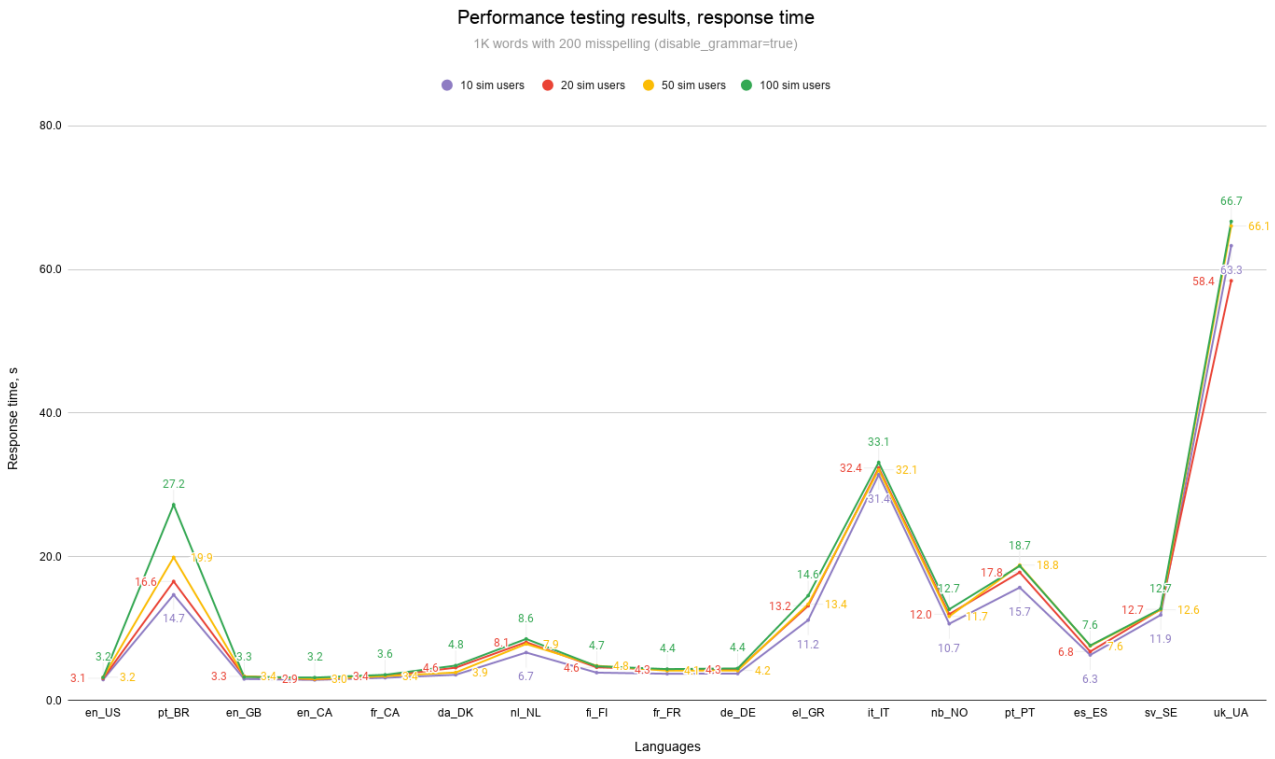
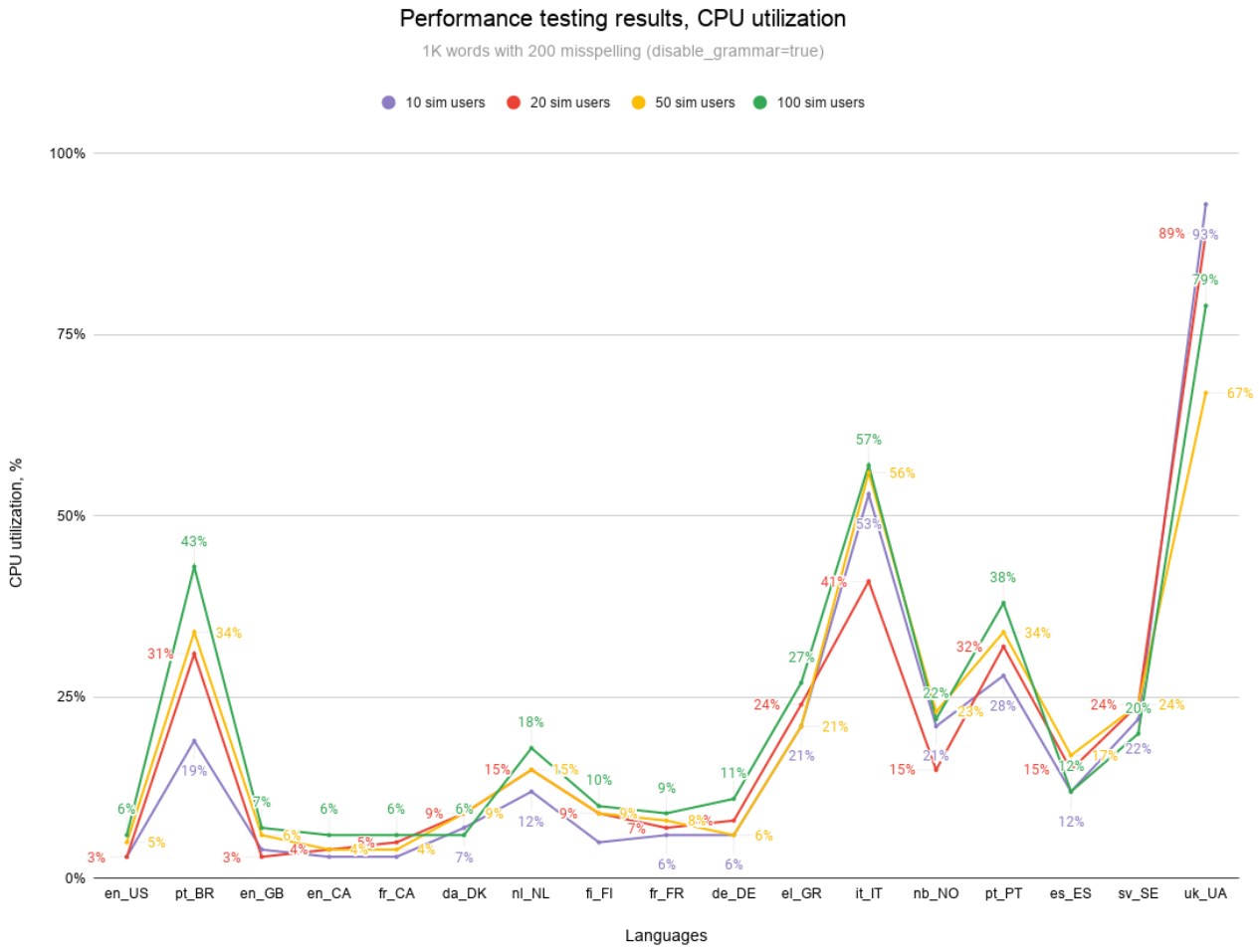


Chart below represents **CPU utilization** results aggregated by language and user tiers when there are **only 200 misspellings** in text of 1K words size.



Response time and CPU utilization (only 50 grammar problems)

Chart below represents **response time** results aggregated by language and user tiers when there are **only 50 grammar problems** in text of 1K words size.

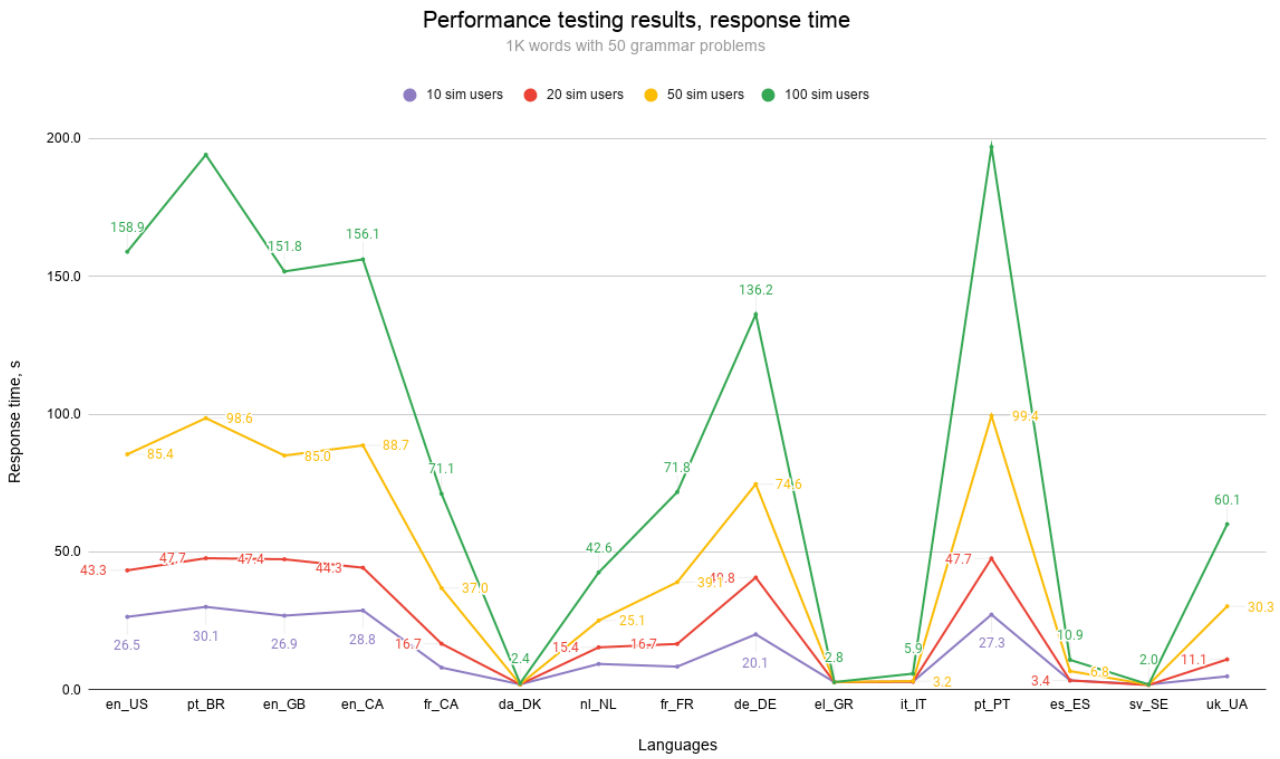
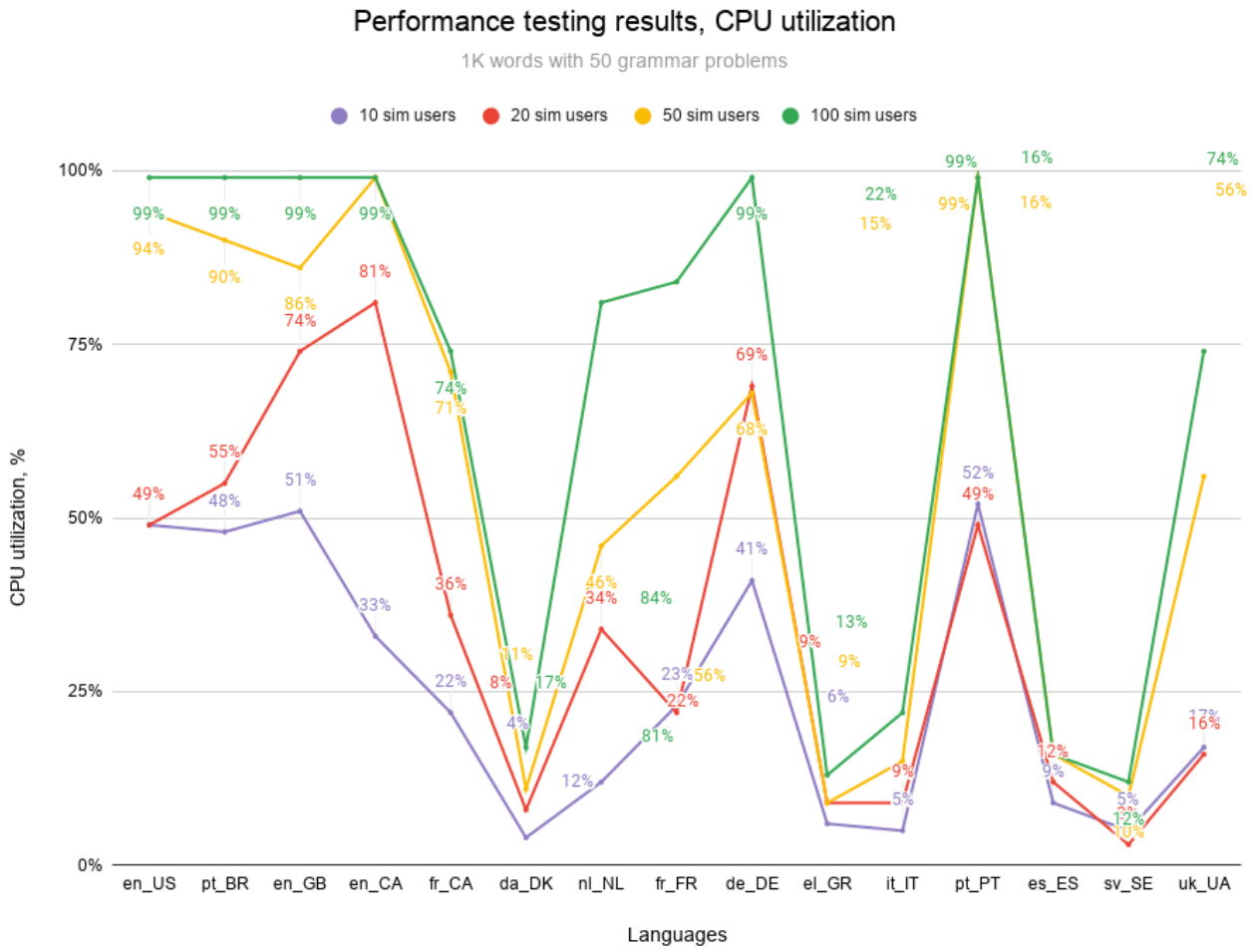


Chart below represents **CPU utilization** results aggregated by language and user tiers when there are **only 50 grammar problems** in text of 1K words size.



Response time and CPU utilization (200 misspellings and 50 grammar problems)

Chart below represents **response time** results aggregated by language and user tiers when there are**200 misspellings** and **50 grammar problems** in text of 1K words size.

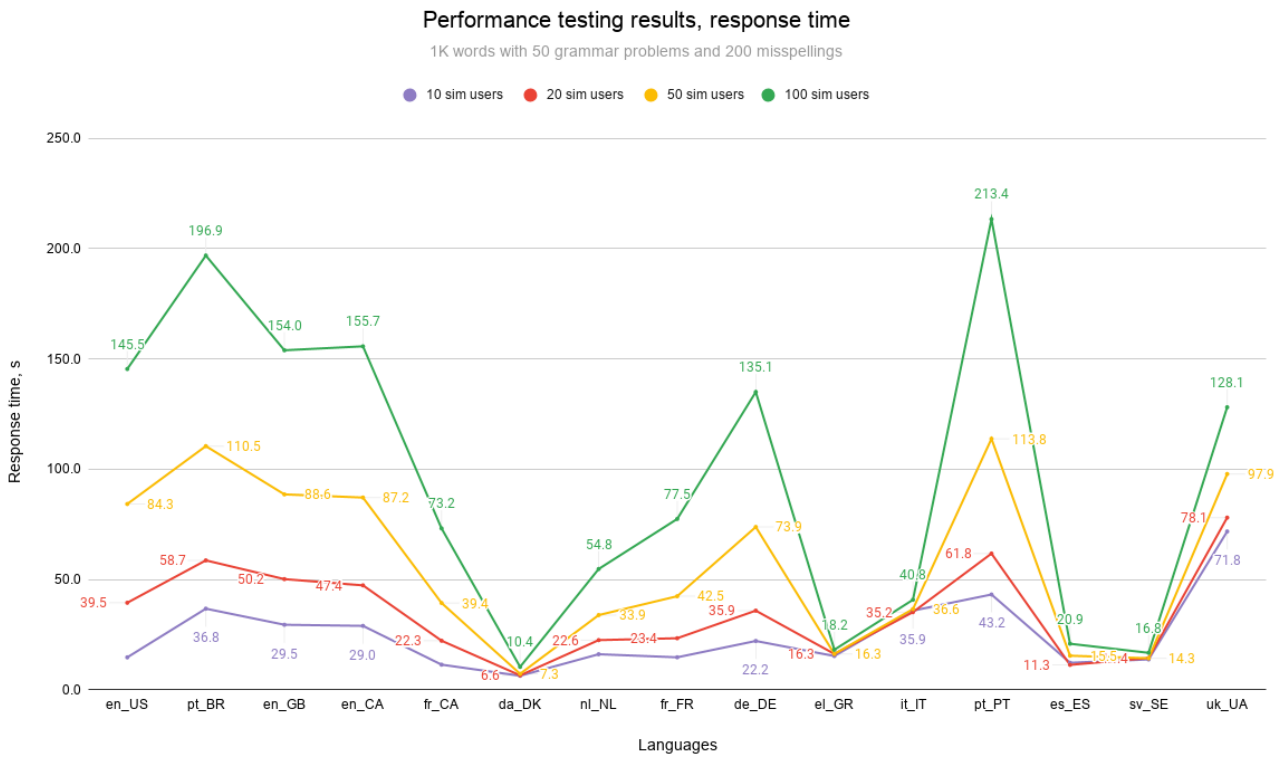
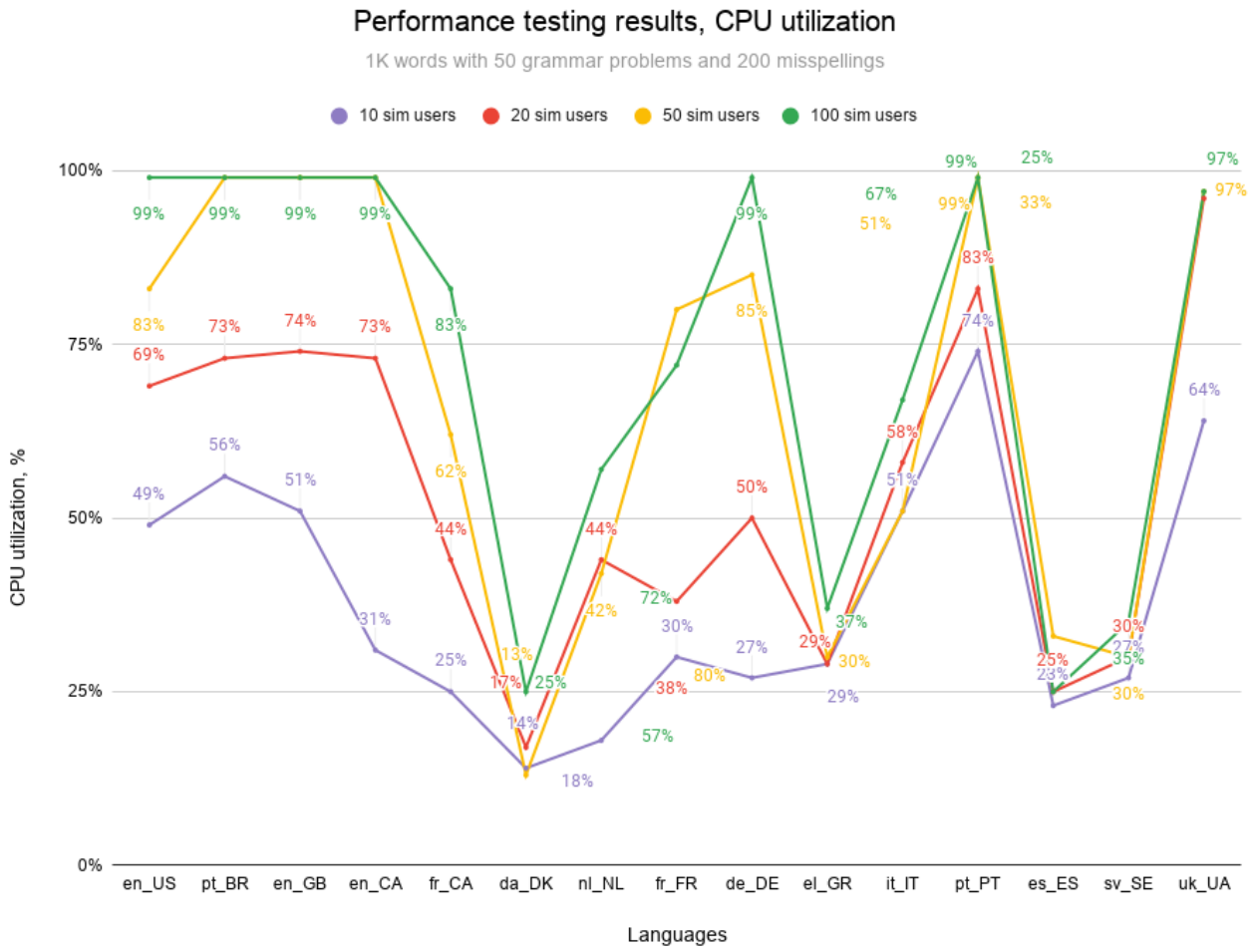


Chart below represents **CPU utilization** results aggregated by language and user tiers when there are **200 misspellings** and **50 grammar problems** in text of 1K words size.



Findings and recommendations

Here are the outcomes and aftermath as well as our advice on hardware and software requirements and notes on performance issues which users may encounter:

> ⚠ Demonstrated results are far away from the reality and real use cases especially if to consider using UI-based products as WProofreader. During testing we used API requests which contain 1K words per request where 20% of words are spelling errors and 5% are for grammar errors. On average a person writing on the second language makes around 4%-5% of errors. This is 5 times less than used during tests. Secondly, a mechanism of requests distribution and size of each requests for UI-based product are being optimized greatly to ensure smooth experience for users and decrease the load on the servers.

- The response time and CPU untilization increase as more simultaneous users are added.
- These two metrics depend a lot on the number of words in the dictionary for spelling check and number of rules for grammar check. The more words and rules are for the language, the higher response time and CPU. For example, Brazillian Portuguese contains over 10 million words in the dictionary, thus, it takes longer to process a request in comparison with other languages. The only exception here is the Ukrainian language, it uses a different spelling check engine which doesn't support multi-threading. As a result it has the worst results.
- The spelling check function in more light weight in comparison with the grammar checking. Having grammar checking enabled more than doubles the response time and increases the CPU utilization.
- One m5 instance can process up to 50 simultaneous users, or simultaneous threads, under given conditions (size of a text and percentage of errors), but when the number of users increases to up 100, it entails 99-100% CPU load and a significant increase of response time. Our recommendation for the case when more users are added and CPU load constantly reaches 100% on the machine:
    - upgrade instance type and add more CPUs to it;
    - add one more machine to distribute the traffic (requests) between two or more machines, for example, using load balance and auto-scalling.

If you have any questions or comments regarding the outlined results, please fill free to contact us.