CS 321

Operating Systems

Fall 2024/2025

Load Balancer Project: Milestone 3 Benchmarking Report and Analysis

---

Defining mathematical models for simulating Traffic patterns.
Overview of the Stress Test's Structure.
Reviewing and Analyzing Performance Results of a Sample Run.

Overview

It is imperative to simulate various types of traffic patterns to evaluate the performance of our Load Balancer to reflect potential real-world scenarios and to test few edge cases that our Balancer might encounter.

In this report, we outline the mathematical models used to calculate and simulate traffic patterns. We will also explain the parameters of these functions. Our wave functions include sine waves, sudden waves and polynomial waves. By selecting these functions, we can observe how our load balancer handles fluctuating loads, sudden spikes in usage and gradual increases respectively.

Subsequent analysis of the plots of our sample run will help in understanding the limitations and the strong points of our implementation of the load balancer, highlighting potential improvements in a few weak fields.

Traffic Models: Mathematical descriptions:

To simulate realistic traffic, three distinct wave functions were used: sine waves, sudden waves and polynomial waves. Each of these functions can model different unique patterns of request arrival allowing us to test the load balancer under varying conditions.

Our selection of variables offers customizability options for testing the load balancer on different base levels, fluctuations and frequencies.

1. Sine waves:

$$F(t) = |Baseline + (Amplitude \times \sin(2\pi \times Frequency \times t))|$$

Where:

t: current time.

$Baseline$: Baseline traffic level (used to simulate continuous background activity).

$Amplitude$: Height of the sine wave.

$Frequency$: Width of the wave: determines the number of waves per unit of time.

Sine waves help us test our Load balancer on smooth, fluctuating number of connections.

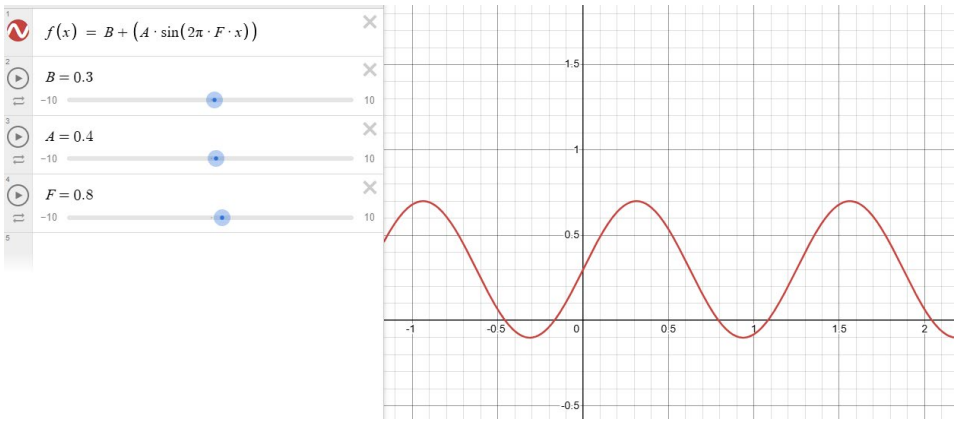


Fig: example of the sine wave demonstrated in Desmos' graphing tool

2. Sudden waves:

$$F(t) = Baseline + Amplitude \times signal\_function(sine\_wave(t))$$

$$sine\_wave(t) = \sin(2\pi \times Frequency \times t)$$

$$Signal\_function(X) = \begin{cases} 1 \; if \; X \geq 0 \\ 0 \; if \; X < 0 \end{cases}$$

Where:

$Baseline$: Baseline traffic level (used to simulate continuous background activity).

$Amplitude$: Height of the spike.

t: current time

$Frequency$: Width of the sine wave

Sinewaves mimic the behavior of square waves, where the value jumps between two binary values . This can simulate extreme and sudden spikes and crashes of connections to our Load Balancer.

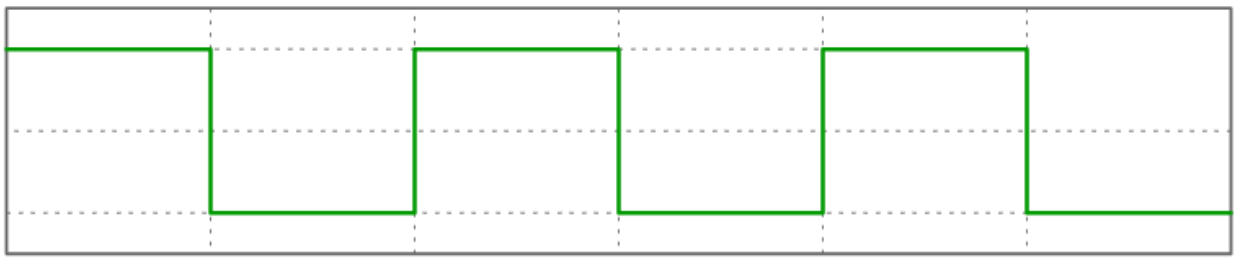


Fig: example of the square wave. Image taken from square wave's Wikipedia page.

3. Polynomial function:

$$f(t) = Baseline + (Amplitude \times (\frac{t}{time\_normalizer}))^{power}$$

Where:

t: current time

$Baseline$: Baseline traffic level (used to simulate continuous background activity).

$Amplitude$: linear scaler, controls the curvature of the parabola.

$time\_normalizer$: Typically equals to the duration of the test. This number scales that graph to the desired time. Its purpose is to introduce a fraction to the equation that counters the amplitude's effect based on the current time.

$power$: power of the polynomial

Polynomial waves aids in simulating the exponential growth of connections to the load balancer.

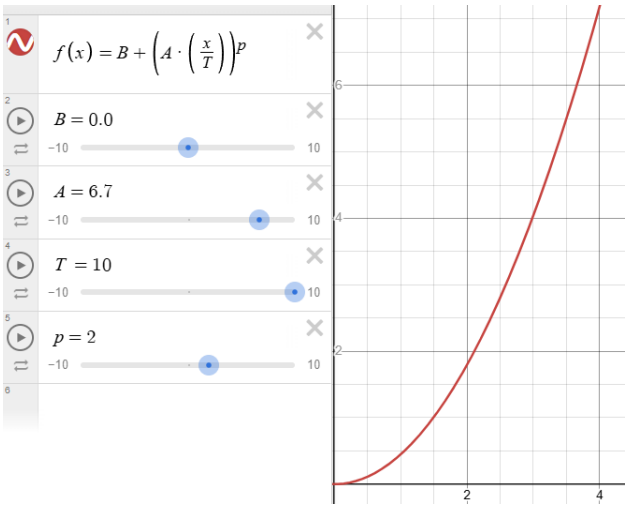


Fig: example of the polynomial wave demonstrated in Desmos' graphing tool

Stress-Testing the Load Balancer

With the traffic models and their parameters defined, we can now implement our stress tester, which should utilize those wave functions and analyze the load balancer's behavior against them.

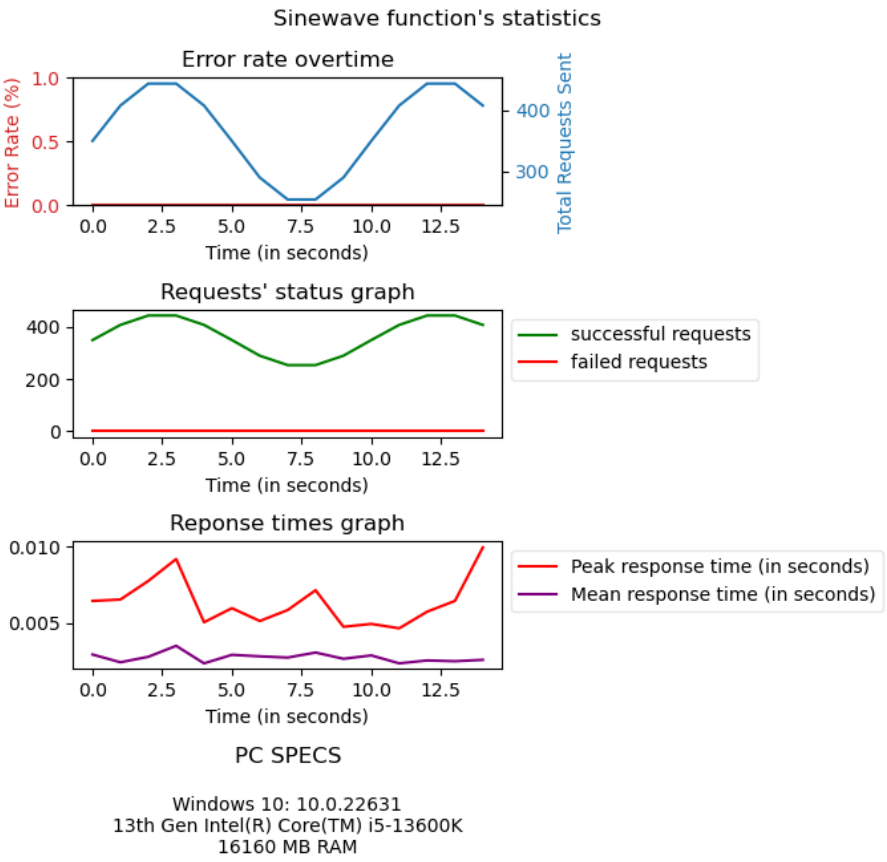
The behavior of our stress tester program is as follows:

1. Sine wave is the first model our program tests our load balancer with. It generates an array indicating the number of requests that should be sent within a second and send said number of requests per second over the duration of the test.
    - The parameters of all our tests were fined tuned to work well on one of our colleagues' PC.
2. After testing the load balancer on the sine wave function, it then prints various statistics regarding the test that was captured every second, those parameters include:
    [1] total_requests_sent: Number of requests sent during that second.
    [2] successful_requests: Number of successful requests handled by the Load Balancer.
    [3] failed_requests: Number of requests the Load Balancer failed to deliver.
    [4] error_rate: Successful to failed requests' ratio.
    [5] mean_response_time: Average response time for all requests during that second.
    [6] peak_response_time: Highest recorded response time for a request on that second.
3. The program repeats the same previous steps against the sudden wave and the polynomial wave functions. It also records their statistics separately.
4. After capturing all the tests' statistics, a csv file is generated for each wave function in addition to a visualized graph for each wave's data outputted as a .png image file.

Sample Run: Results and Analysis:

The following plots illustrate statistics regarding the traffic patterns generated by each wave function and their corresponding test data.

1. Sine wave plot:



Test parameters:

- Duration: 15 seconds
- Baseline: 350
- Frequency: 0.1
- Amplitude: 100

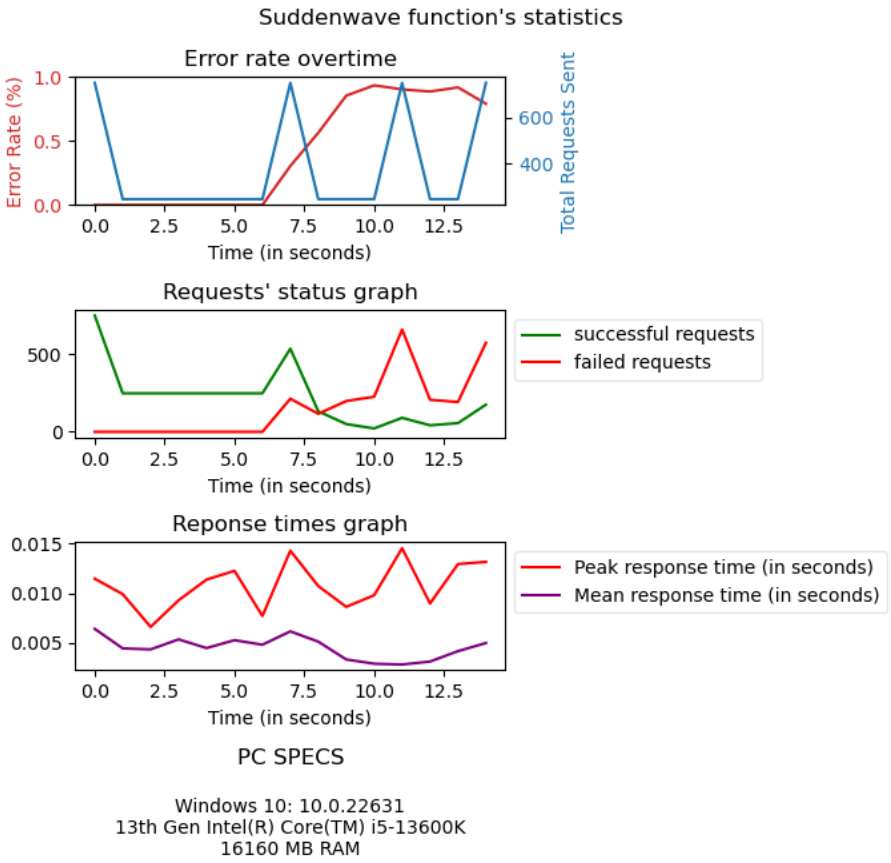
Analysis:

The algorithm demonstrated excellent performance against this wave function. It maintained a perfect score for successful requests. The maximum recorded response time is 10 milliseconds, which was the best peak response time recorded throughout the entire benchmark.

We concluded that our load balancer can handle fluctuating number of requests very well, maintaining great stability and overall performance.

sudden DDOS like

2. Sine wave plot:



Test parameters:

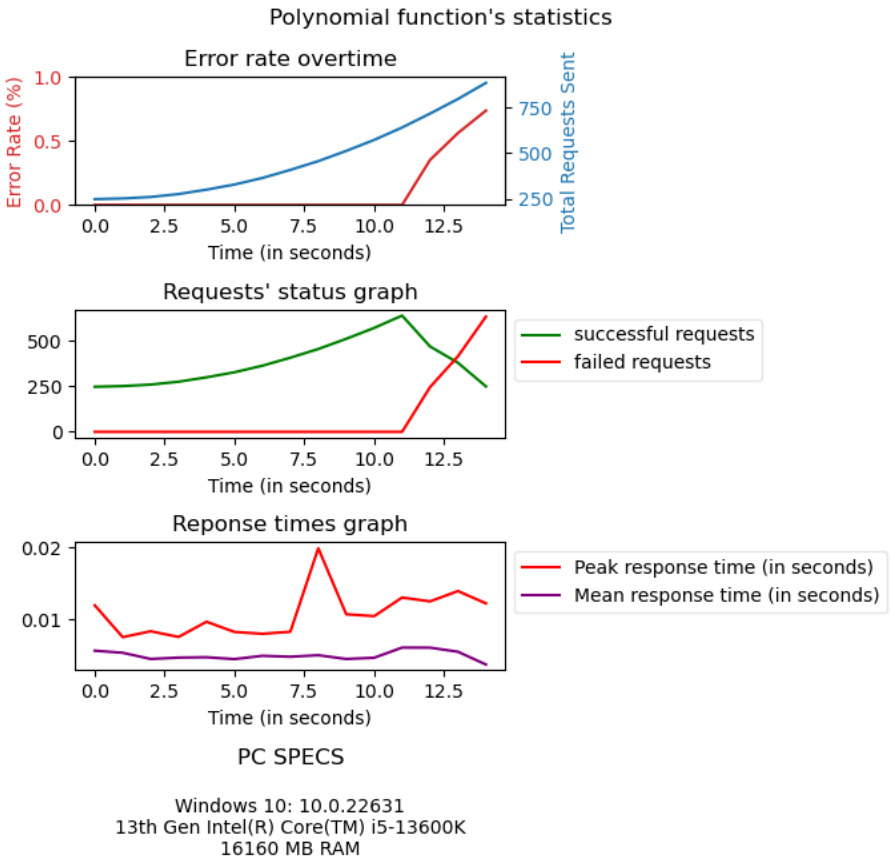
- Duration: 15 seconds
- Baseline: 250
- Frequency: 9
- Amplitude: 500

Analysis:

The algorithm showed volatile performance patterns upon encountering the first spike. A relationship between failed requests and spike's timeframe was noticed, indicating the model's incapability to handle extreme sudden traffic conditions. Response times were also less predictable. Its peaks and crashes do not correlate with the spike's timeframes.

The erratic pattern of this wave function suggests potential stability issues and demands top priority for any future planned improvements.

3. Polynomial wave plot:



Test parameters:

- Duration: 15 seconds
- Baseline: 250
- Frequency: 9
- Amplitude: 500

Analysis:

The algorithm displayed decent performance at the beginning, but a sharp spike of failed requests was noticed at the 11 second mark. The exponential growth of the total requests' throughput contributed to the degrading performance. Response times remain relatively stable but showed a spike around the 8 second mark. This peak response time was the worse one recorded during this test, but the difference was negligible overall.

The polynomial implementation works well until it hits a breaking point, after which the performance rapidly degrades.

Stress-Test conclusions:

- Sinewave implementation fits the natural behavior of a production environment and demonstrates the best overall performance.
- Exceptional or erratic production environment behaviors contributed negatively to our algorithm's performance, showing an increase in dropped requests, but still maintaining a relatively stable response times.
- The volatility of sudden spikes in addition to the exponential growth of requests for polynomial wave showed us that, while the algorithm handles moderate traffic conditions reasonably well, it fails to maintain stable performance during high traffic bursts or rapid increases in request volumes.