

# Performance and Cost Analysis of Web Application Deployment on Raspberry Pi versus Enterprise Host

Chandradeep Chowdhury  
cchowdhu@calpoly.edu

Eben Sherwood  
esherwoo@calpoly.edu

June 10, 2023

## 1 Introduction

This project aims to analyze the performance, resource utilization, and cost aspects of deploying a web application on a Raspberry Pi device compared to an enterprise-level hosting environment. We evaluate the response time, resource usage, and associated costs involved in running the web application on both platforms as well as identify the bottlenecks. We hope that this analysis will aid new budget oriented web developers determine the optimal hosting platform for their next web application.

## 2 Experimental Setup

### 2.1 Application

For this project, we will use a Monitor Recommendation web application designed by one of the co-authors. This application accepts two types of request:

- Get-Course
- Find-Monitor

The application is structured as follows:

```
/
/recommender
/crash-course
/api/monitor-recommendations
```

The Get-Course request simply returns a “Monitors 101 Course Page” that is designed to help users unfamiliar with basic monitor terminology. In this project we are mainly interested in the Find-Monitor request.

The Find-Monitor request from /recommender calls the API endpoint at /api/monitor-

recommendations and displays up-to 8 monitors. The request sends a JSON object with 25 fields.

/ corresponds to the Landing page of the website.

The UI of the application was designed using React@18 and the recommender API was made using Flask@2.2.3, a python backend framework. The application runs in a Gunicorn Web Server Gateway Interface (WSGI) server. The application does not have any dedicated database server and uses a csv file to store the monitor information.

On average, this website gets 50 users per day.

### 2.2 Test Environment

In this project, we are comparing the same application hosted on two different platforms:

- Raspberry Pi 3B+ with a 4-core CPU and 1GB memory running a clean installation of Raspbian Linux 11, a flavor of Debian. No other user level applications except gunicorn are allowed to run.
- Render.com enterprise hosting on starter tier with 0.5 vCPU and 512MB Memory running on Debian Linux 10. Only one instance is enabled and autoscaling is turned off.

The load is generated using `wrk`, a modern HTTP benchmarking tool, from a local computer connected to a 500 Mbps internet connection. `wrk` reports the distribution of the total response time and throughput. To measure the CPU and memory utilization, we use a script that calls the Linux `top` command at fixed intervals and then calculates the mean, on both the render.com server and the RPi.

### 2.3 Test Configuration

To load test this application on the two different platforms we are sending a custom JSON object to the API directly. The static UI is simple and we assume that it does not contribute significantly to the total

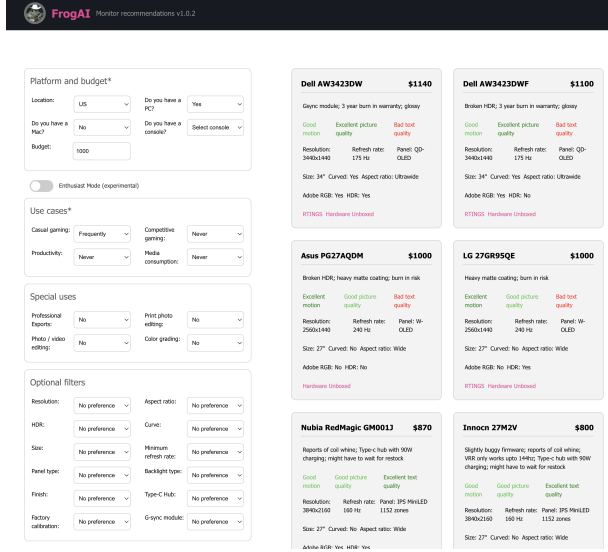


Figure 1: FrogAI recommendation website

latency. We created a simple Lua script that defines the request parameters:

```
wrk.method = "POST"
wrk.body = [[ "country": "US",
"pcGpu": "yes", "consoles": "no",
"mac": "no", "budget": 1000, "mode":
"basic", "casual": "imp", "comp":
"imp", "text": "imp", "media": "imp",
"persistence": "not", "response":
"not", "contrast": "not", "brightness":
"not", "volume": "not", "sharp":
"not", "subpixel": "not", "esports":
"not", "edit": "no", "print": "no",
"grade": "no", "aspect": "nopref",
"curve": "nopref", "size": "nopref",
"res": "nopref", "minRR": "nopref",
"panel": "nopref", "backlight":
"nopref", "hdr": "nopref", "finish":
"nopref", "calibrated": "nopref",
"hub": "nopref", "module": "nopref"
]]
wrk.headers["Content-Type"] =
"application/json"
```

For the initial test, the load generator is configured to use two threads with 5 concurrent connections at a time each to send requests for 1 minute to the two platforms. This amounts to 10 connections open at a time which is a reasonable expected load for this application. We are also using a timeout duration of 5s as anything above that is considered unacceptable for this application. Each test is run 5 times.

System	Stats	Mean
RPI 3B+	$R$	2.18s
	$X$ (req/s)	4.25
Starter	$R$	1.11s
	$X$ (req/s)	8.78
	$U_{CPU}$	0.452

Figure 2: Initial results

System	Conns.	$R$	$X$ (req/s)	$U_{CPU}$
RPI 3B+	2	0.47s	4.26	0.248
	10	2.20s	4.46	0.266
	20	4.32s	4.46	0.263
	50	10.27s	4.38	0.266
	250	30.21s	4.31	0.265
Starter	2	0.25s	8.14	0.450
	10	1.10s	8.99	0.486
	20	2.20s	8.86	0.459
	50	5.35s	8.89	0.500
	250	21.54s	8.82	0.500

Figure 3: Variable load results

Then, we run a series of tests with varying loads to determine the primary bottleneck in this system. We set the timeout to 100s in this case. Each test is run 3 times.

## 3 Results

We summarize the results of the initial experiments in 2. Memory utilization remains constant around 100MB on both systems during idle periods as well as stress tests. There were no timeouts and all responses were received within 5s on both platforms.

We summarize the results of the variable load tests in 3.

### 3.1 Performance Analysis

Python has a Global Interpreter Lock (GIL) that prevents applications from using more than 1 core using multiple threads. Further the Raspberry Pi has 4 CPU's and `top` reports all 4 cores busy as 1.0. Based on these two facts, we suspect that the utilization being consistently around 0.25 on the RPi indicates that only core was loaded 100% the entire time. For the Starter instance the utilization does not go above 0.5 usually as the current plan only allows that much.

It performs worse than the starter instance while having no ability to scale up and being sensitive to power outages. However, we still see some use cases

where it might be the choice, especially for hobbyist applications.

### **3.2 Cost Analysis**

The Render.com starter instance costs \$7.00 per month as long as we do not go over the free build minutes and bandwidth usage which is not a concern for this application. The Raspberry Pi 3B+ has initial hardware cost of \$35 and a power usage cost of approximately \$0.67 per month in California. Over a long period of time, it is clearly cheaper than the enterprise hardware.

## **4 Future Work**

## **5 Conclusion**