# Week 5. Research Assignment 2



Attempt 1	~	In Progress
		NEXT UP: Submit Assignment



#### **Unlimited Attempts Allowed**

#### ∨ Details

Performance benchmarks comprise an important component of this course. We draw examples from academic research and industry studies. This assignment involves the construction of a performance benchmark across popular Go and/or Python backend frameworks.

### Management Problem

Managers of a technology startup are evaluating alternative software stacks for the company's web applications. They know that Go will serve the company's needs for backend web servers and applications, but they are uncertain about which backend web framework to use, if any. Also, there are a number of data scientists in the firm that are more familiar with Python than Go, and they have been promoting the use of a Python backend framework, such as Django, Flask, or FastAPI,

Searches of the web point to numerous backend Go web frameworks, providing utilities that complement the Go standard library, including routing, session management (with cookies), security, and system logging. As well, there is a contingent of engineers arguing that the Go standard library is sufficient and that no Go web framework be used for application development.

Echo (https://echo.labstack.com/) is described as a minimalist but extensible Go framework. Fiber (https://docs.gofiber.io/), which draws on fastHTTP (https://github.com/valyala/fasthttp) rather than net/http from the Go standard library, is known for its fast performance. And Gin (https://gin-gonic.com/) is the most popular Go web framework. Golang Dojo offers a playlist of introductory videos on Go Web App development (https://www.youtube.com/watch?

v=OsyqKRZyz4o&list=PLve39GJ2D71yyECswi0IVaBm\_gbnDRR9v).

Scalability and performance are key in selecting a web application development environment. The firm's data scientists suggest that comprehensive performance benchmarks be run across four web frameworks being considered. For Go, they are considering None (standard Go library only), Echo, Fiber, and Gin. For Python, they are considering Django, Flask, and FastAPI.

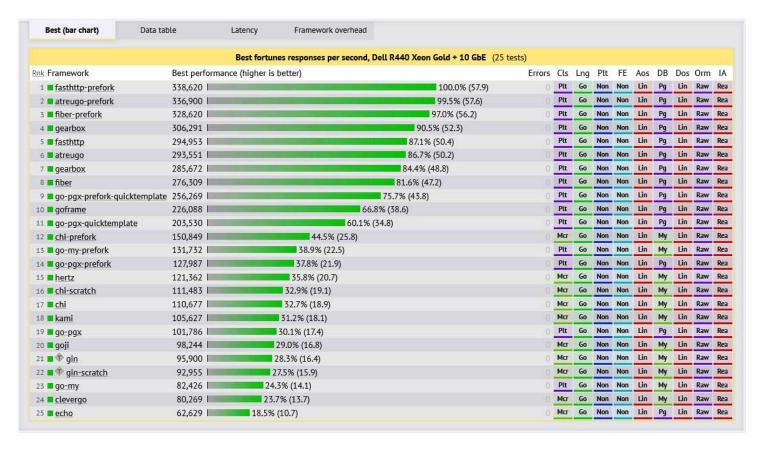
### Lessons from Industry Benchmarks

TechEmpower → (https://www.techempower.com/) runs performance benchmarks across hundreds of backend web frameworks, reporting results → (https://www.techempower.com/benchmarks/#hw=ph&test=fortune&section=data-r22) online and documenting research methods in a public GitHub repository → (https://github.com/TechEmpower/FrameworkBenchmarks). Results across Go and Python backend frameworks show great variability both in throughput and latency.

TechEmpower's throughput test is called **Fortunes**:

In this test, the framework's ORM is used to fetch all rows from a database table containing an unknown number of Unix fortune cookie messages (the table has 12 rows, but the code cannot have foreknowledge of the table's size). An additional fortune cookie message is inserted into the list at runtime and then the list is sorted by the message text. Finally, the list is delivered to the client using a server-side HTML template. The message text must be considered untrusted and properly escaped and the UTF-8 fortune messages must be rendered properly.

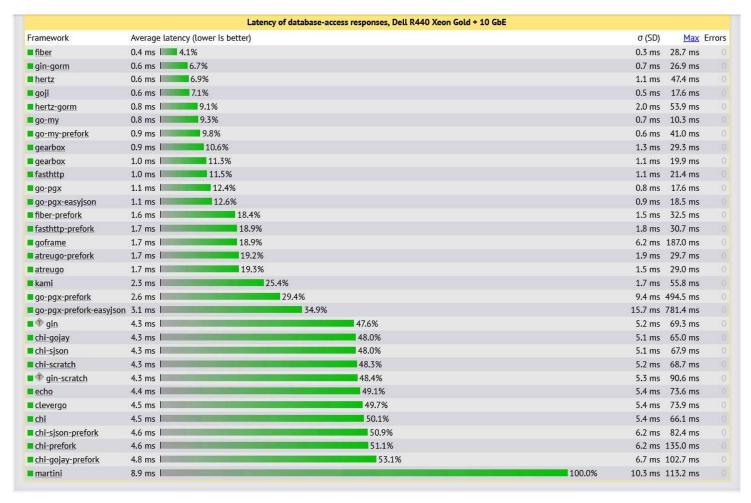
Here are the throughput results across Go web frameworks:



TechEmpower uses a single-query latency test:

In this test, each request is processed by fetching a single row from a simple database table. That row is then serialized as a JSON response.

Here are TechEmpower's latency results across Go web frameworks:



## Assignment Requirements

Assume the role of a company data scientist. Design and conduct a benchmark study across the two web framework options: Go None, Go Echo, Go Fiber, Go Gin, Python Django, Python Flask, or Python FastAPI.

Examine both throughput and latency in querying an SQLite database, creating at least two queries for both throughput and latency. Define appropriate response measures for throughput and latency.

For database queries, consider using an SQLite database. It is not necessary to use an object-relational mapper for this study. But if an object-relational mapper is used, ensure that it is used across each of the selected web framework options.

Conduct a Monte Carlo performance benchmark, running each query task at least one hundred times, obtaining response distributions across the runs, and computing average response measures across the runs for each treatment and each task. Construct tables and figures summarizing study results.

The README.md file of your public GitHub repository for this assignment should provide

- A description of repository folders and files
- Information about the research design (treatment conditions and data)
- Instructions about how to run the benchmark study
- Tables and figures summarizing benchmark results
- · Recommendation for management: Based on performance indices alone, which web framework should the firm use?

## Assignment Deliverables

- Link to your GitHub repository with the .git extension.
- · Include all benchmark code.
- Document research results in listings, tables, and figures.
- The Readme.md file of your GitHub repository should review your methods and results.

# Grading Guidelines (50 total points)

- Benchmark research design (10 points)
- Benchmark software (10 points)
- Documentation of benchmark results (10 points)
- Interpretation and recommendation to management (10 points)
- Exposition (10 points)

Enter Web URL			
	https://		
	<	>	
(https://canvas.nor	thwestern.edu/courses/221184/modules/items/3143486)	(https://canvas.northwestern.edu/courses	/221184/modules/item