MODULE 1 – THE BIG PICTURE

IT 207 – IT Programming

GEORGE MASON UNIVERSITY

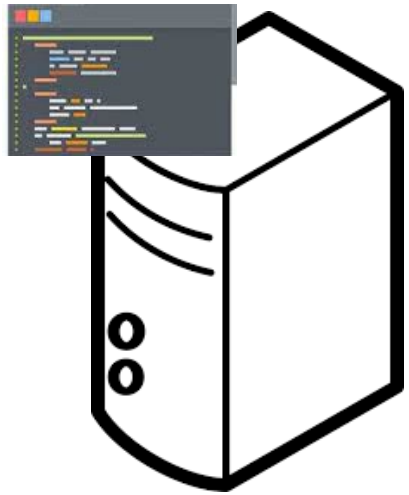
Lecture 1 – Outline

- What is a Web application and how does it work?
- What does HTTP and URL stand for?
- Types of client-server applications
- Server concurrency and server design
  - Nodejs Performance and I/O Scaling Problem
- Node.js Basic Architecture

WEB APPLICATION OVERVIEW

Web Applications

- A web application is
  - A **distributed** application that uses a **Client-server** architecture
  - Its code is stored and runs on a **server**.
  - Interacts with the user through a **browser**

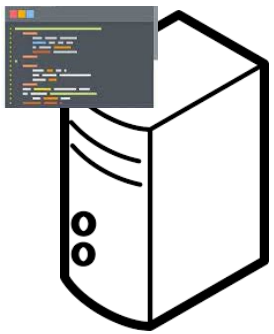


Server

Internet

Web Applications Cont'

- A web application is
  - A **distributed** application
  - **Client-server** program
  - Executes locally in the user **browser**
  - Its code is stored on a **server**.
- Examples:
  - Web email,
  - Shopping carts,
  - Online banking
  - Etc.



Server

Internet

Web Applications – how does it work?

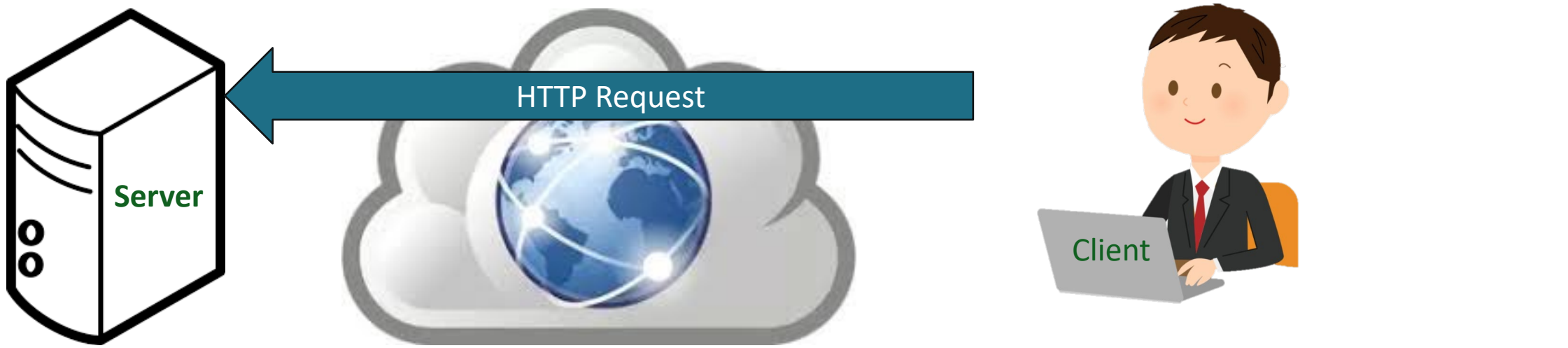
- The user types a URL in the browser

Uniform resource locator

Web Applications – How does it work?

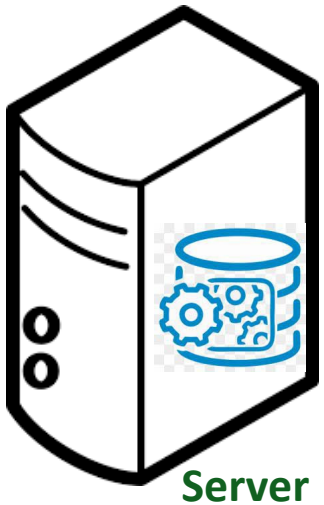
- The user types a URL in the browser
- The browser sends a request to the server using HTTP

Hyper Text Transfer protocol



Web Applications – How does it work?

- The user types a URL in the browser
- The browser sends a request to the server using HTTP
- The server processes the request.



Server

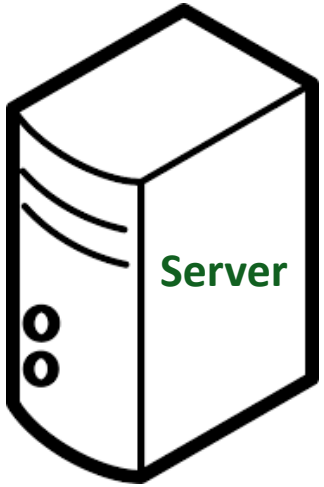
Internet

Web Applications – How does it work?

- The user types a URL in the browser
- The browser sends a request to the server using HTTP
- The server processes the request
- The servers sends back the requested web resource(s) in an HTTP response

Web Applications – How does it work?

- The user types a URL in the browser
- The browser sends a request to the server using HTTP
- The server processes the request
- The servers sends back the requested web resource(s) in an HTTP response
- The browser logs the response in its window



Internet

HTTP, URL & TYPES OF CLIENT-SERVER APPS

Hyper Text Transfer Protocol

- Protocol:
  - set of rules that defines how data is formatted, sent, and received between computers on the network.

Hyper Text Transfer Protocol

- **Protocol:**
  - set of rules that defines how data is formatted, sent, and received between computers on the network.
- **HTTP:**
  - The main protocol used for data exchange on the web.
  - It was invented alongside HTML to create the first interactive, text-based web browser.
  - It is a client-server protocol that defines two types of messages:
    - requests, and
    - responses.

HTTP Messages

- **Request:**
  - HTTP requests are sent by the client to trigger an action on the server.
  - HTTP defines several request methods
    - POST
    - GET
    - DELETE
    - PUT
- **Responses:**
  - HTTP responses are sent back by the server to the client carrying the results of the action triggered by the request.
  - The action triggered depends on the request method

Uniform Resource Locator

- **Resource:**
  - A resource can be an HTML page, a CSS document, or an image.
- A URL has the form

- URL is is first sent to a DNS to get the equivalent server address before sending out the request on to the Internet.

Types of Client-Server Applications

- Data Intensive
  - Applications that make an intense usage of data in all its forms.
  - Vast majority of web applications.
  - Require frequent access to data storage to either store data or retrieve data.

Types of Client-Server Applications

- Data Intensive
  - Applications that make an intense usage of data in all its forms.
  - Vast majority of web applications.
  - Require frequent access to data storage to either store data or retrieve data.
- CPU Intensive
  - Applications that depend heavily on the CPU to perform intensive computations.
  - Rarely perform I/O operations
  - Usually done offline

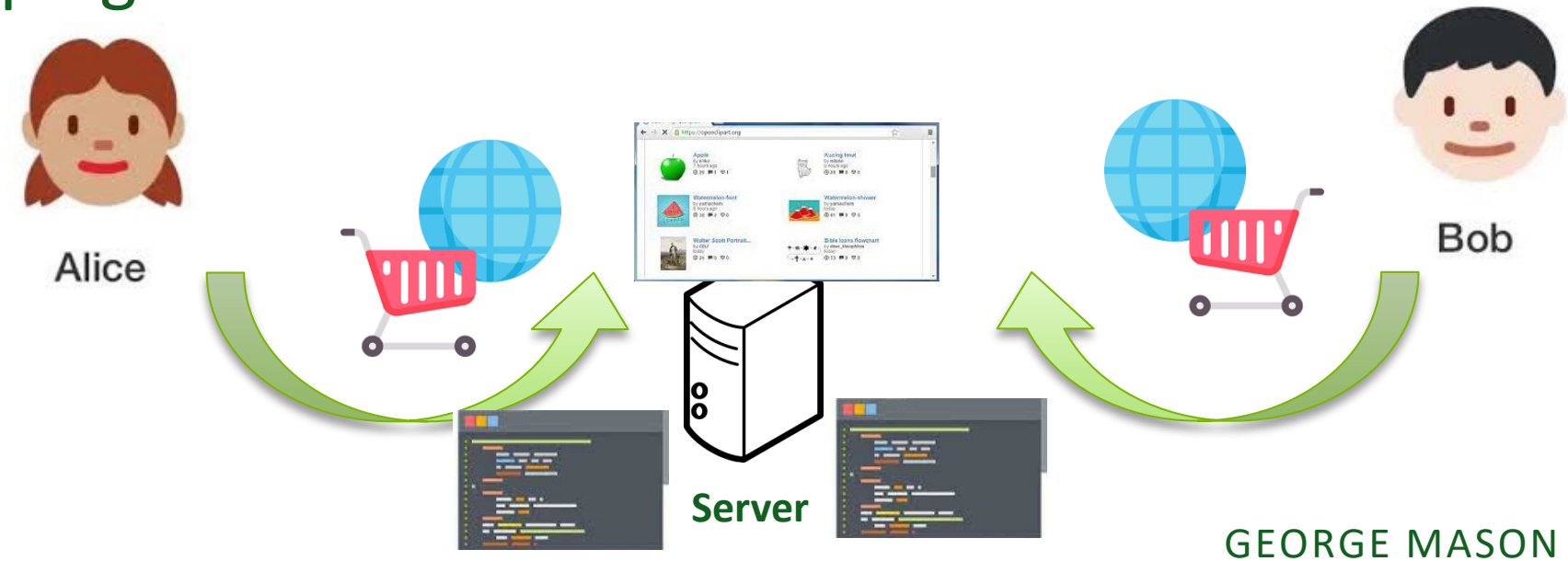
CONCURRENT SERVER DESIGNS

Server Concurrency & Server Design

- In a Client-Server architecture, a server is a program that provides services to its clients through a request/response communication protocol.
  - The server is required to serve many clients concurrently.
  - The server runs multiple instances of its programs (processes).

Server Concurrency & Server Design

- In a Client-Server architecture, a server is a program that provides services to its clients through a request/response communication protocol.
  - The server is required to serve many clients concurrently.
  - The server runs multiple instances of its programs (processes).
- Shopping cart Scenario



Server Concurrency & Server Design

- In a Client-Server architecture, a server is a program that provides services to its clients through a request/response communication protocol.
  - The server is required to serve many clients concurrently.
  - The server runs multiple instances of its programs (processes).

Process:
As instance of a *running* program

Concurrency:
A technique in which two or more processes execute on the same CPU in an *interleaved* fashion

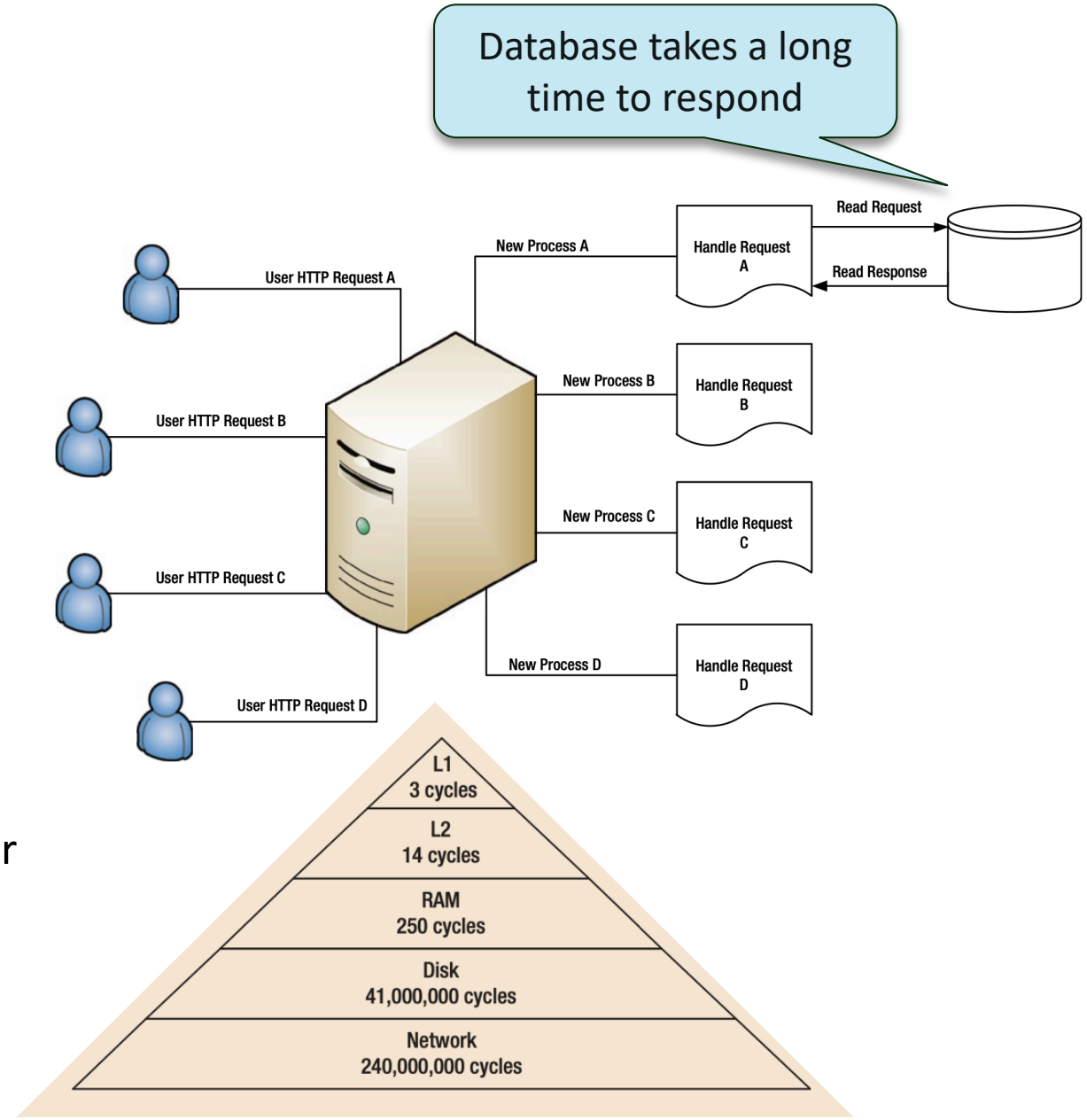
Server Concurrency & Server Design

- In a Client-Server architecture, a server is a program that provides services to its clients through a request/response communication protocol.
  - The server is required to serve many clients concurrently.
  - The server runs multiple instances of its programs (processes).
- Server concurrency can be done through
  - Forking
  - Threading
  - Multiplexing

APPROACHES TO CONCURRENCY FORKING

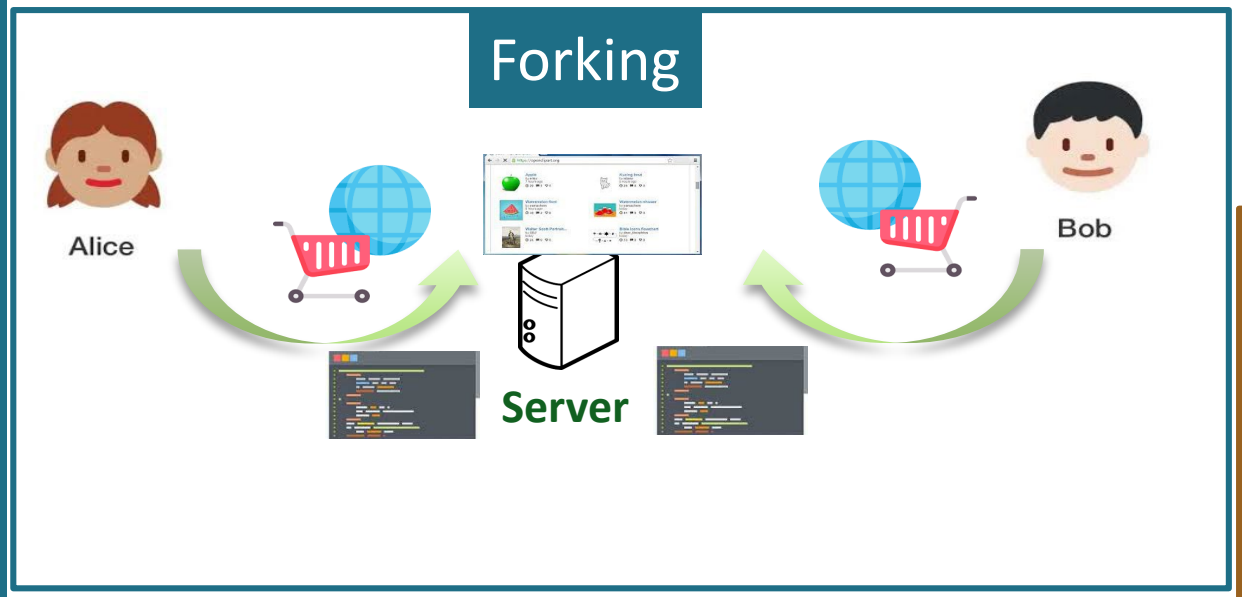
Server Design

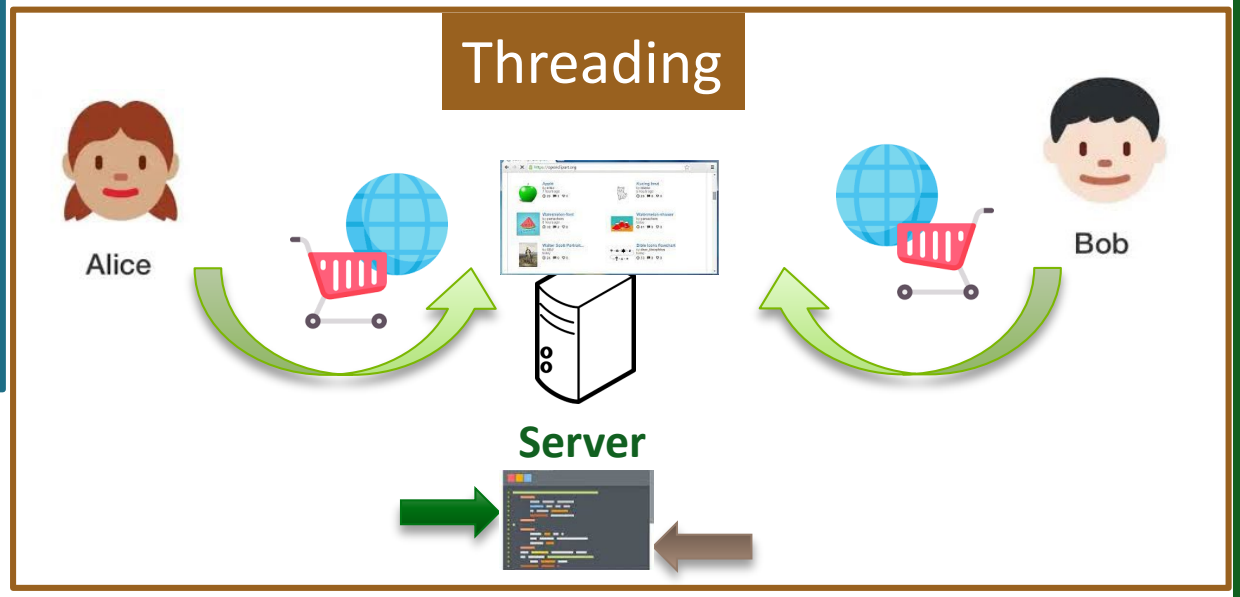
- Forking
  - A new process is created For each request received.
  - Used by traditional web servers to serve multiple requests concurrently.
- Forking is an expensive
  - Each new process is allocated memory and takes a share of the CPU time.
- Forking is inefficient
  - During database access, a process takes up CPU and memory while idling and waiting for the database response.



Server Design – Threads vs. Processes

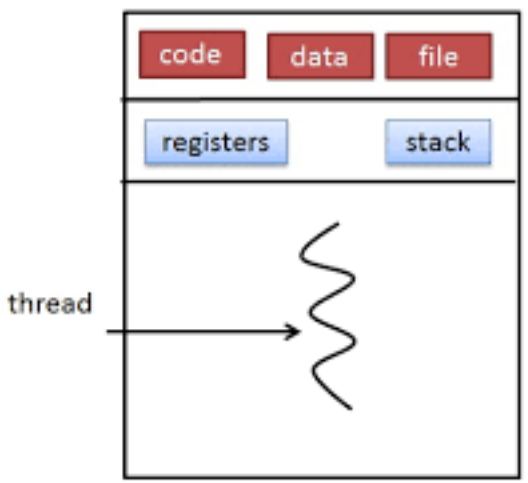
A thread is a sequence of code that is executed within the scope of the process.



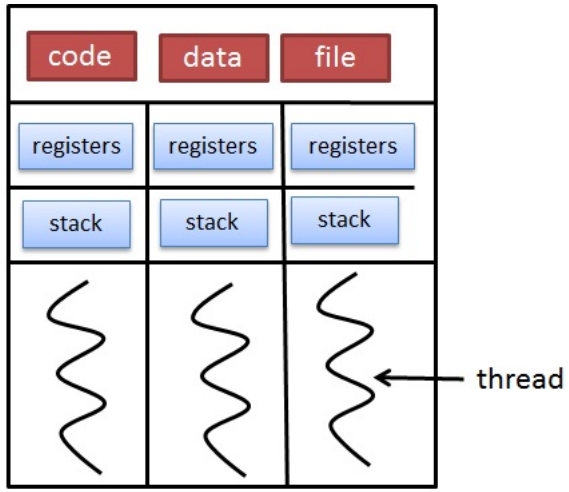


Server Design – Threads vs. Processes

- Threading:
  - The server creates a thread pool within a single process.
  - Threads within the process share the code, but each thread will have a private memory space for its data.
  - Thread are called light processes since they are less demanding with respect to memory requirements.
  - Threads still need a share of the CPU time (cycles) to run.

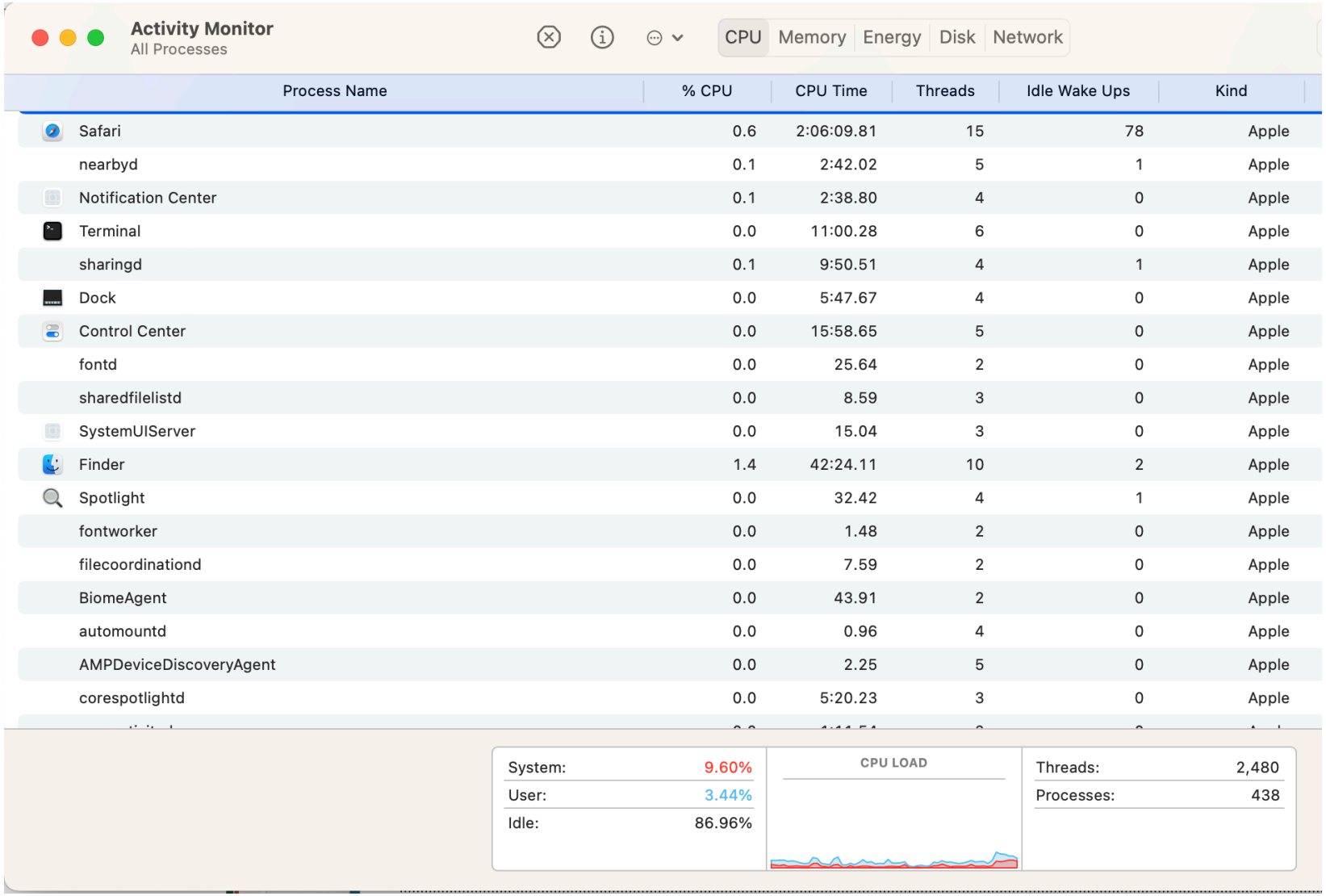


Single-threaded Process



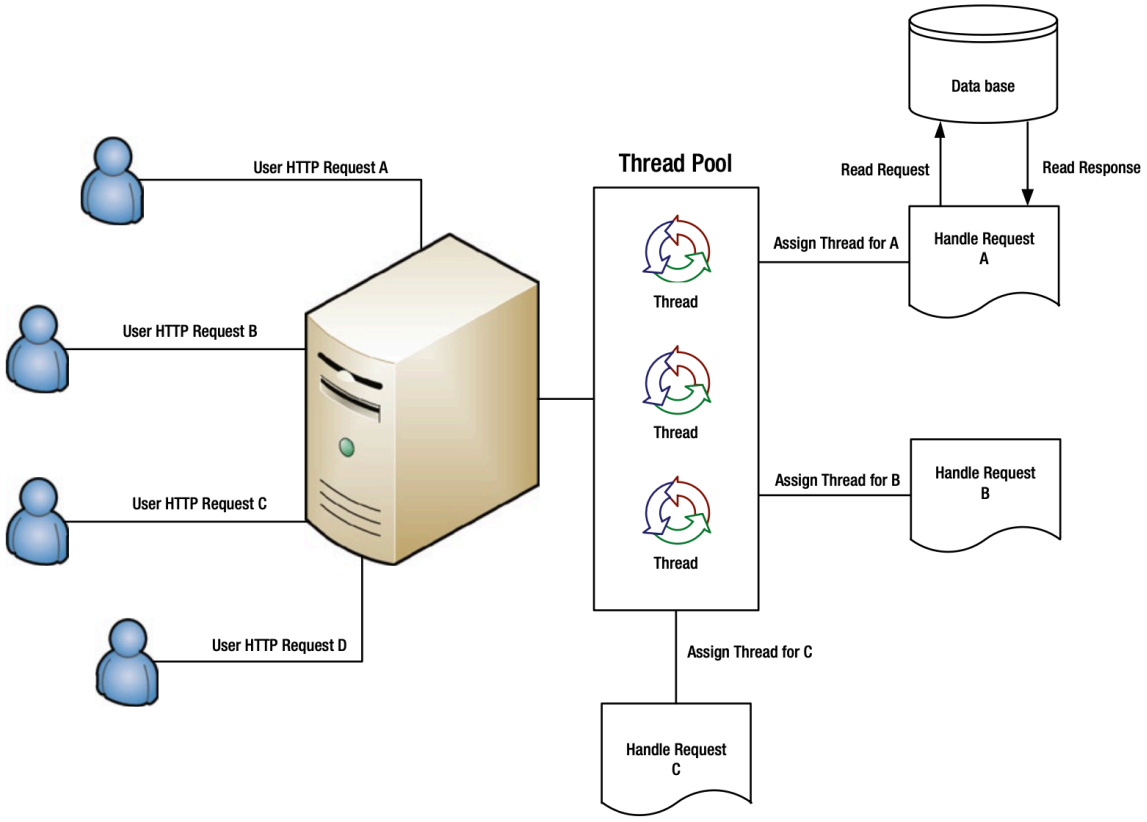
Multithreaded Process

Activity Monitor – Threads vs Processes



Server Design – Threading

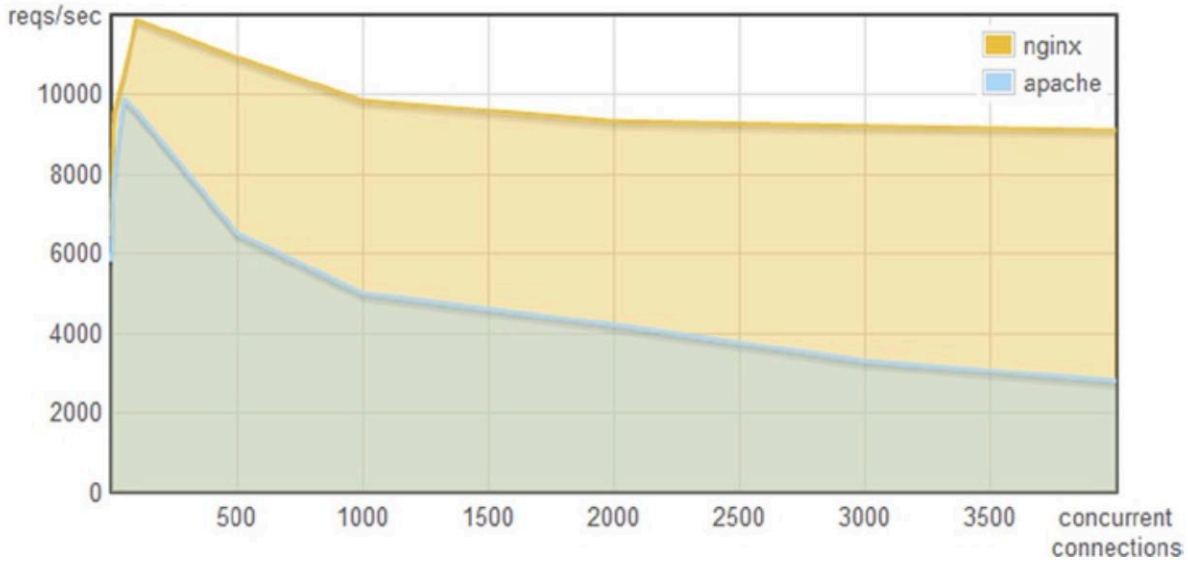
- Threading
  - Modern servers use a thread from a thread pool to serve each request.
  - The thread is reserved for the request in the entire duration that the request is being handled
- Threading is inefficient
  - Threads waste CPU cycles while idling and waiting for the database response.



Server Design – Multiplexing

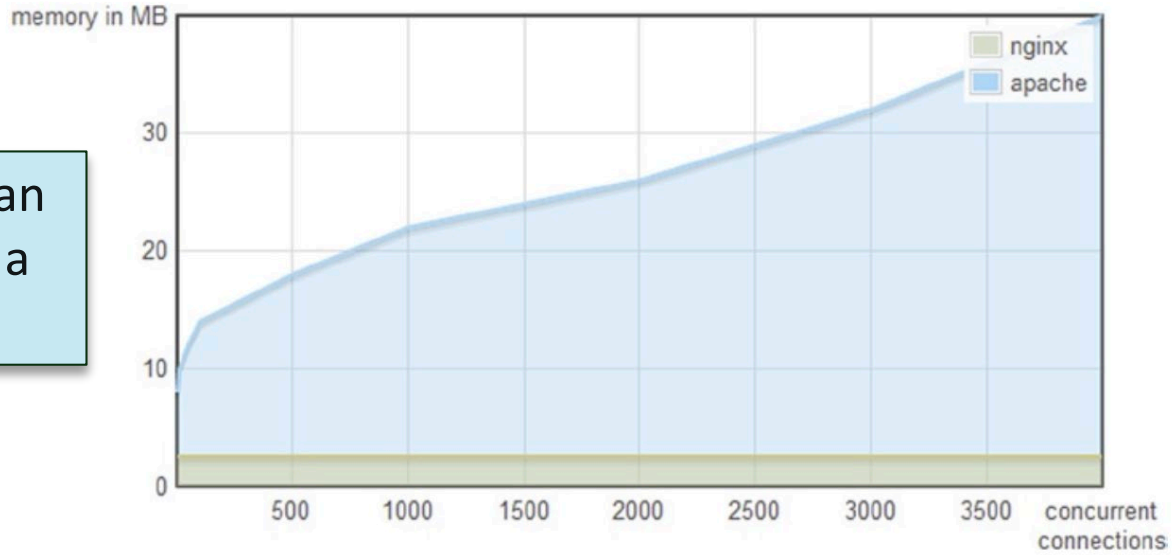
- Multiplexing:
  - The server has one single process with one main thread to serve client's requests
  - Requests arriving at the server are stored in a queue and are served by the single thread in turn in a first come first served fashion.
  - If a request initiates a blocking operation (reading a file, querying DB, etc.…), the operation is passed over to be fulfilled in the background and the main thread is free to serve another request.
  - Once the blocking operation is complete the result will be sent back to the main thread.
  - The main thread sends the result to the client in a response message

SERVER DESIGN - THREADING VS MULTIPLEXING



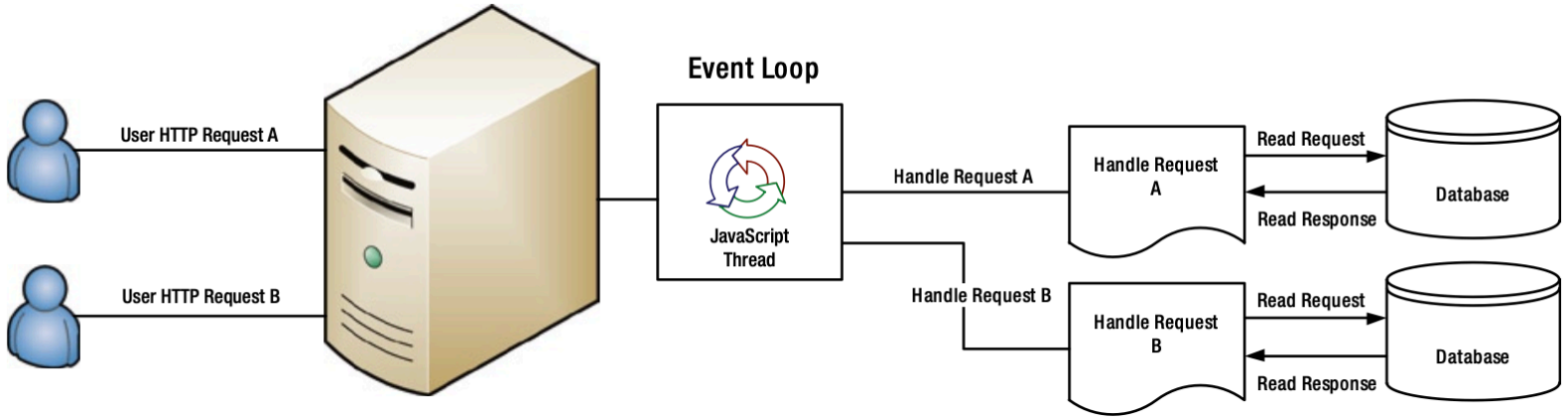
The NGINX server is a multiplexing server
The APACHE server is a threading server

When compared to APACHE, the NIGINX server can serve more concurrent connections while having a very low memory footprint



NODEJS PERFORMANCE

- Nodejs builds on the multiplexing principle used by the NIGINX and utilizes the single thread executing in JavaScript to have an extremely low-memory footprint when operating a web server



NODEJS ARCHITECTURE

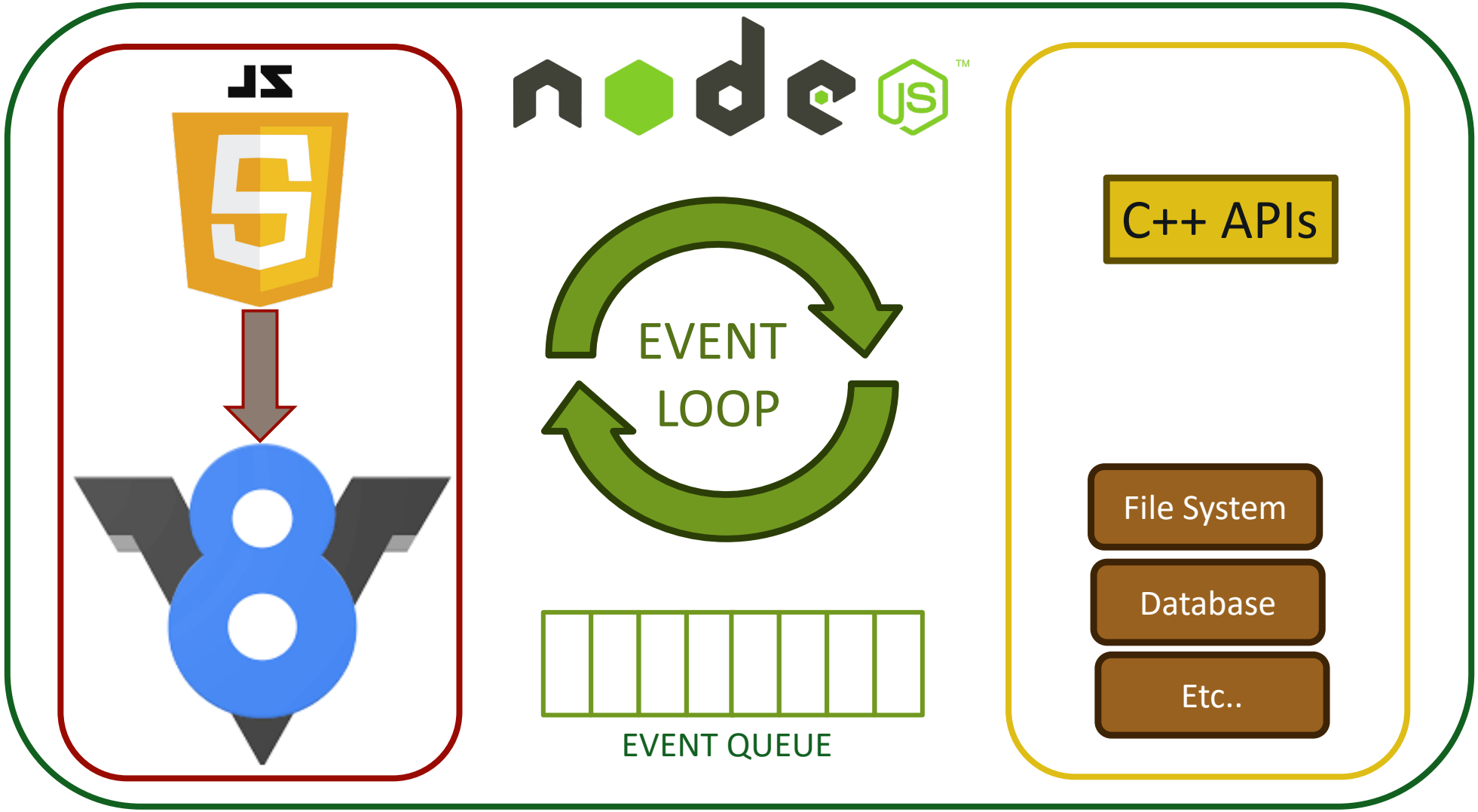
NODE.JS

- As stated on the Official Node page

> "Node.js is a single threaded, open-source, cross-platform JavaScript runtime environment, built on top of the Google Chrome V8 JavaScript engine. Node.js is mainly used to create web servers - but it's not limited to just that."

- Open Source: Nodejs code is made available for use or modification as users or other developers see fit.
- Cross platform: Compatible to run on different operating systems and different computer architectures.
- Runtime Environment: The environment in which a program or application is executed. It's the hardware and software infrastructure that supports the running of a particular codebase in real time.
- JavaScript Engine: a program converts JavaScript code into machine language that can be executed on the computer

Node.js Basic Architecture



The Event loop in Action

```
console.log('This should come First');
console.log('This should come second')
console.log('This should come last');
```

```
This should come First
This should come second
This should come last
```

```
console.log('This should come First');
setTimeout(()=>{console.log('This should come second')}, 10);
console.log('This should come last');
```

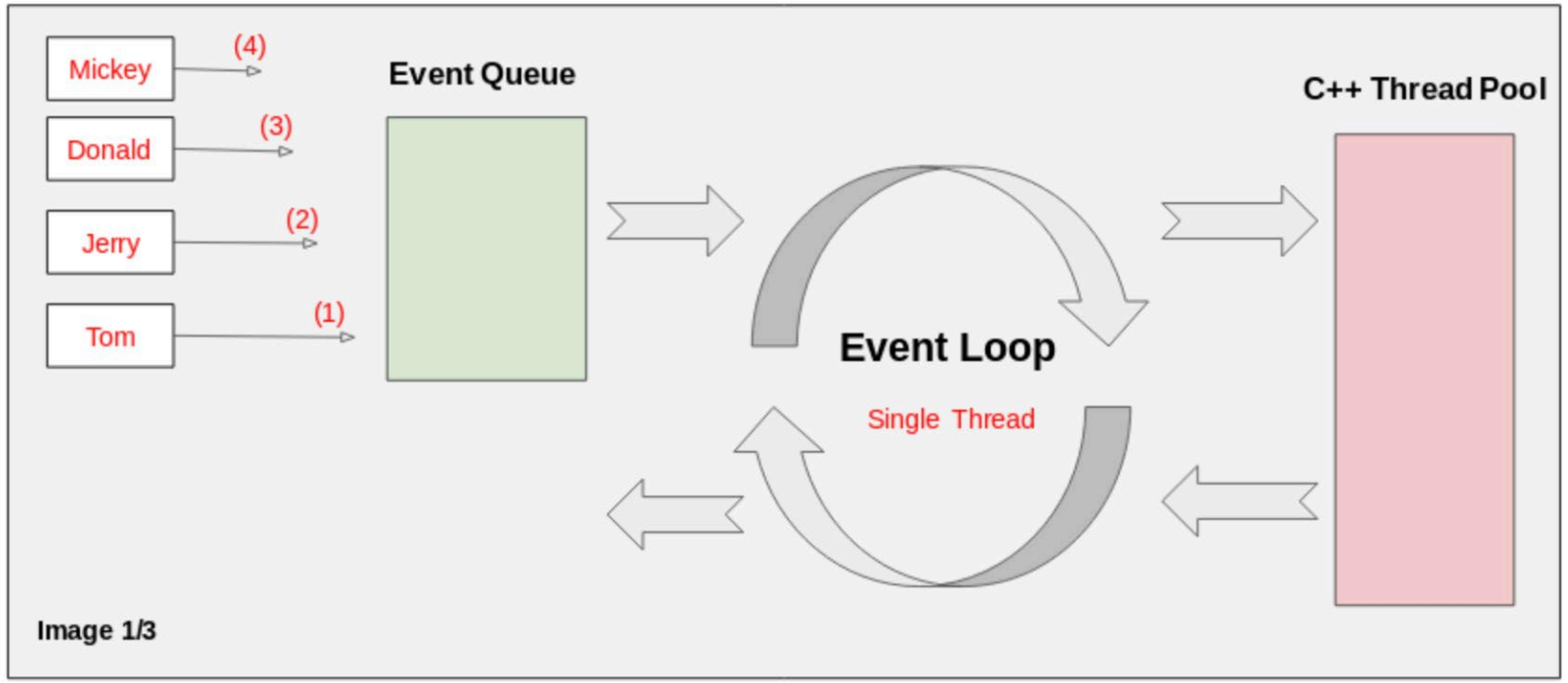
What is the expected output?

```
This should come First
This should come last
This should come second
```

NODEJS OPERATION – AN EXAMPLE

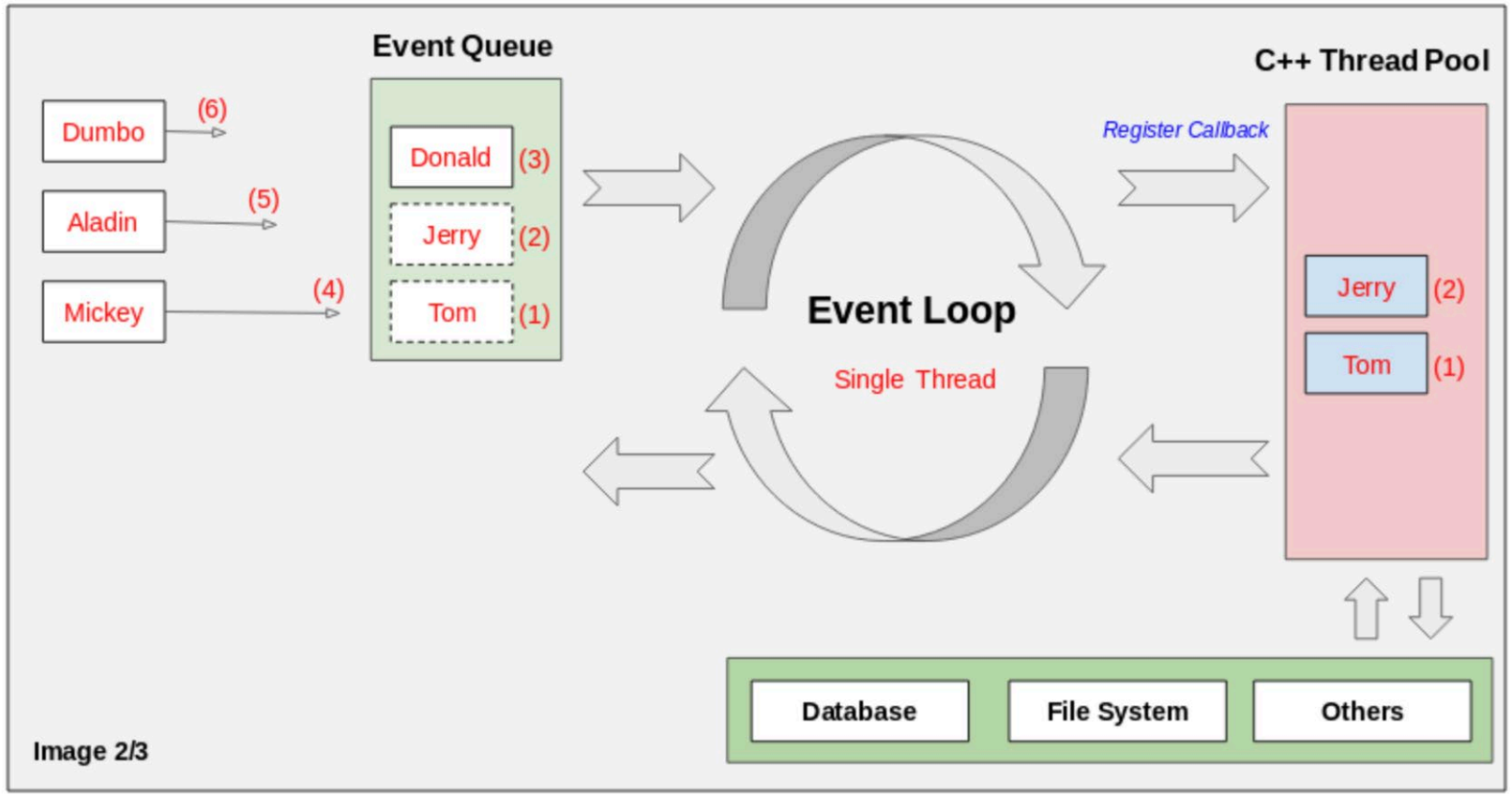
Example: Nodejs Operation

- Clients send requests to the Nodejs Server.
- Each request received at the Nodejs server is considered as an event.
- Requests get queued into the event queue and are handled by the always-running event loop in a first come first serve order.



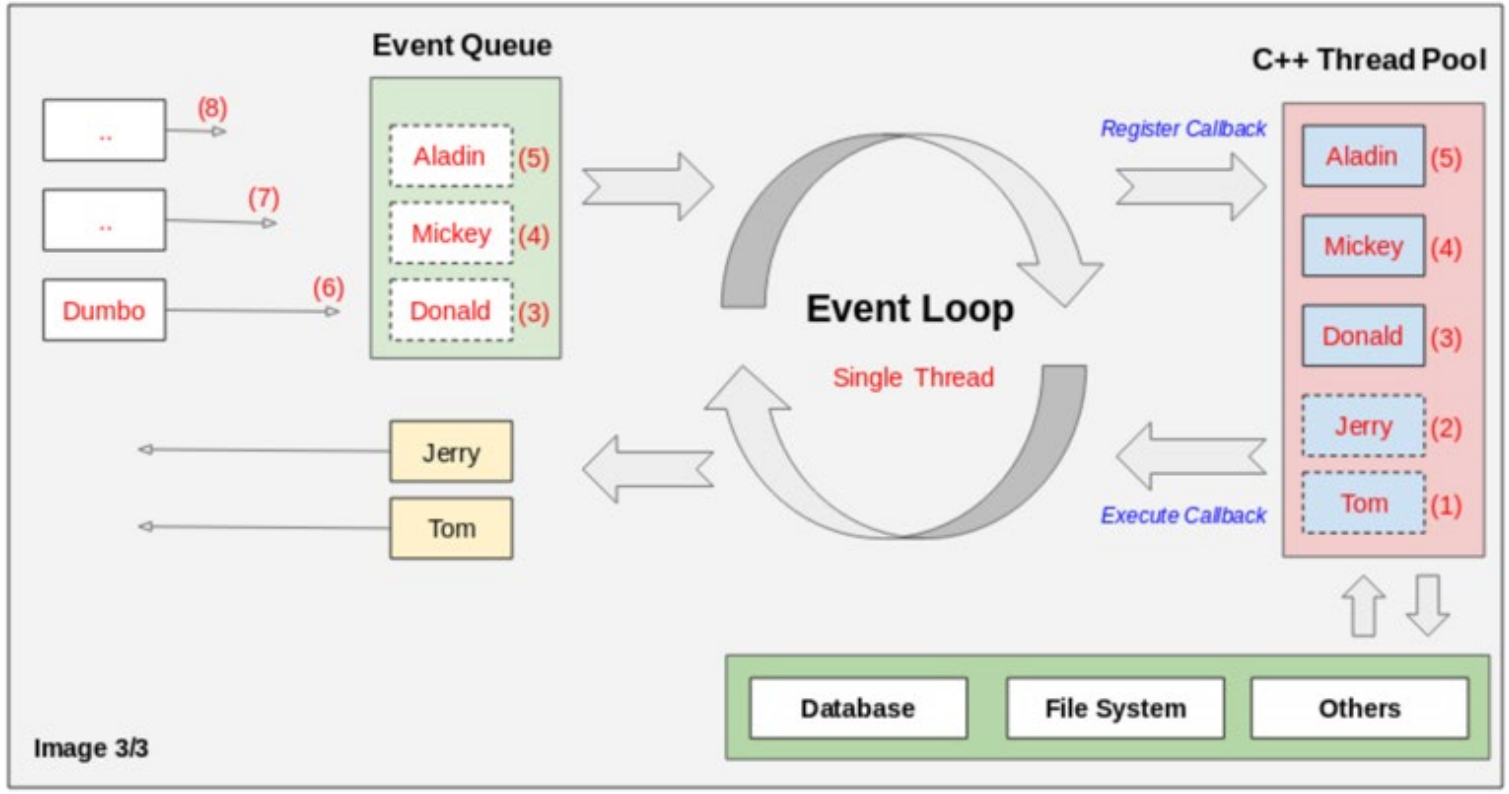
Example: NodeJS Operation

- All requests have a callback function associated with them.
- If a request requires a task that needs time to complete, the event loop passes the task to the C++ API in the background.



EXAMPLE: NODEJS OPERATION

- When the C++ API completes the task, it returns the result through the callback function
- The Event loop executes the callback function which returns the result in a response to the client.



SUMMARY

- Web application is a distributed applications that is composed of two parts; the Client and the Server.
- Client and Server use the HTTP communication protocol for exchanging messages.
- HTTP defines two type of messages requests and responses.
- There are different approaches to support server concurrency.
- To achieve concurrency for data intensive applications multiplexing showed better performance over forking and threading
- Nodejs is a runtime environment for JavaScript built over Google's V8 JS engine
- Nodejs builds on the multiplexing principle and utilizes the single thread executing in JavaScript to to create highly performant and efficient web servers.

REFERENCES

1. Syed, Basarat Ali. Beginning Node.js. Berkeley, CA: Apress. Web.
2. **NodeJS Event Loop Overview**, https://o7planning.org/11951/nodejs-event-loop
3. **Server-side website programming**, https://developer.mozilla.org/en-US/docs/Learn/Server-side