NODE JS

Aruna S

Department of
Computer Science and Engineering

NODE JS

NodeJS Introduction

S. Aruna

Department of Computer Science and Engineering

- Node.js is an open source, cross-platform runtime environment built on

  Google Chrome's JavaScript Engine (V8 Engine).

- Node.js was developed by Ryan Dahl in 2009 and its latest version is v12.18.3

- Node.js is a platform built on chrome's JavaScript runtime for easily building

  fast and scalable network applications.

- Node.js uses an event-driven, non-blocking I/O model that makes it

  lightweight and efficient, perfect for data-intensive real-time applications

  that run across distributed devices.

- Node.js is open source, completely free, and used by thousands of

  developers around the world.

- Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

- Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

Node.js = Runtime Environment + JavaScript Library

When you create websites with PHP for example, you associate the language with an HTTP web server such as Apache or Nginx. Each of them has its own role in the process:
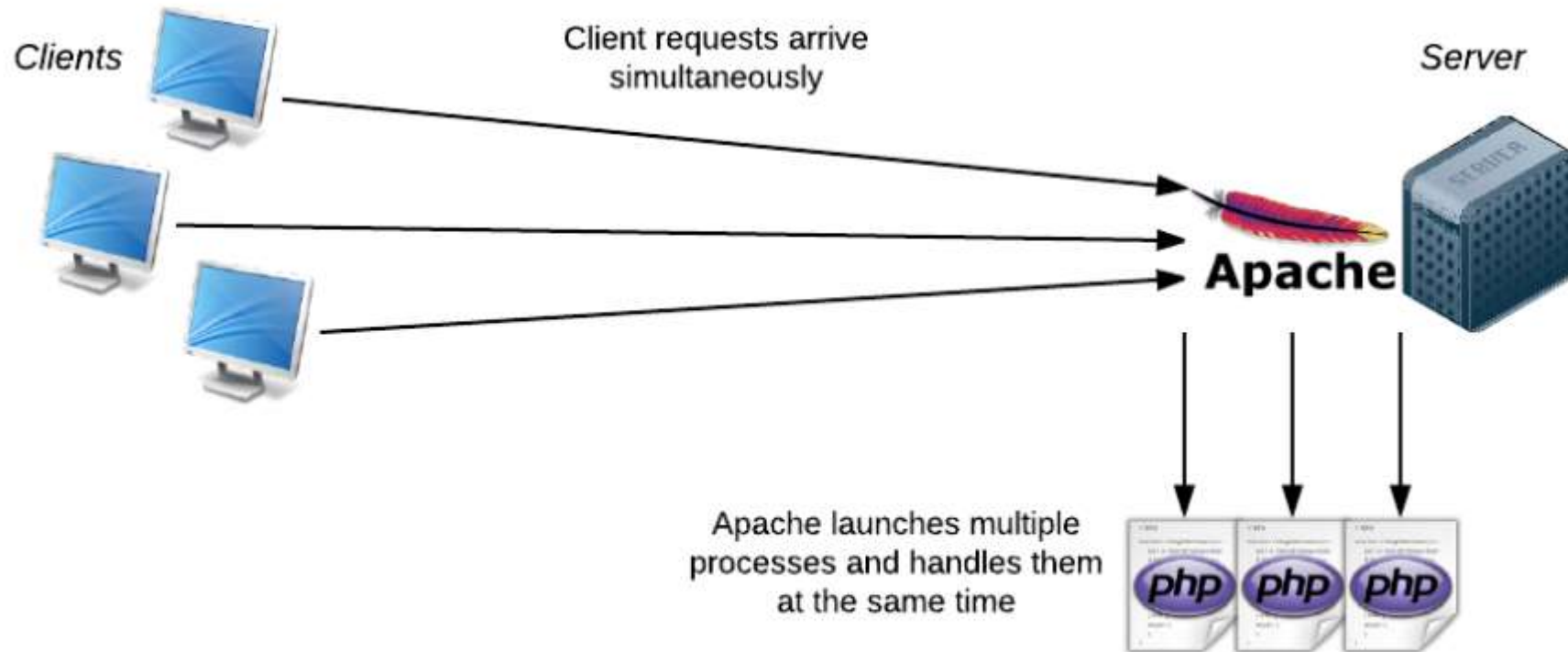
- Apache manages HTTP requests to connect to the server. Its role is more or less to manage the in/out traffic.

- PHP runs the .php file code and sends the result to Apache, which then sends it to the visitor.

As several visitors can request a page from the server at the same time, Apache is responsible for spreading them out and running different *threads* at the same time.

Each thread uses a different processor on the server (or a processor core)

Clients

Client requests arrive simultaneously

Server

Apache

Apache launches multiple processes and handles them at the same time

php php php

The Apache server is multithread

- Node.js doesn't use an HTTP server like Apache. In fact, it's up to us to create the server! Isn't that great?

- Unlike Apache, Node.js is **monothread**. This means that there is only one process and one version of the program that can be used at any one time in its memory.

But I thought that Node.js was really fast because it could manage loads of requests simultaneously. If it's monothread, can it only perform one action at a time?
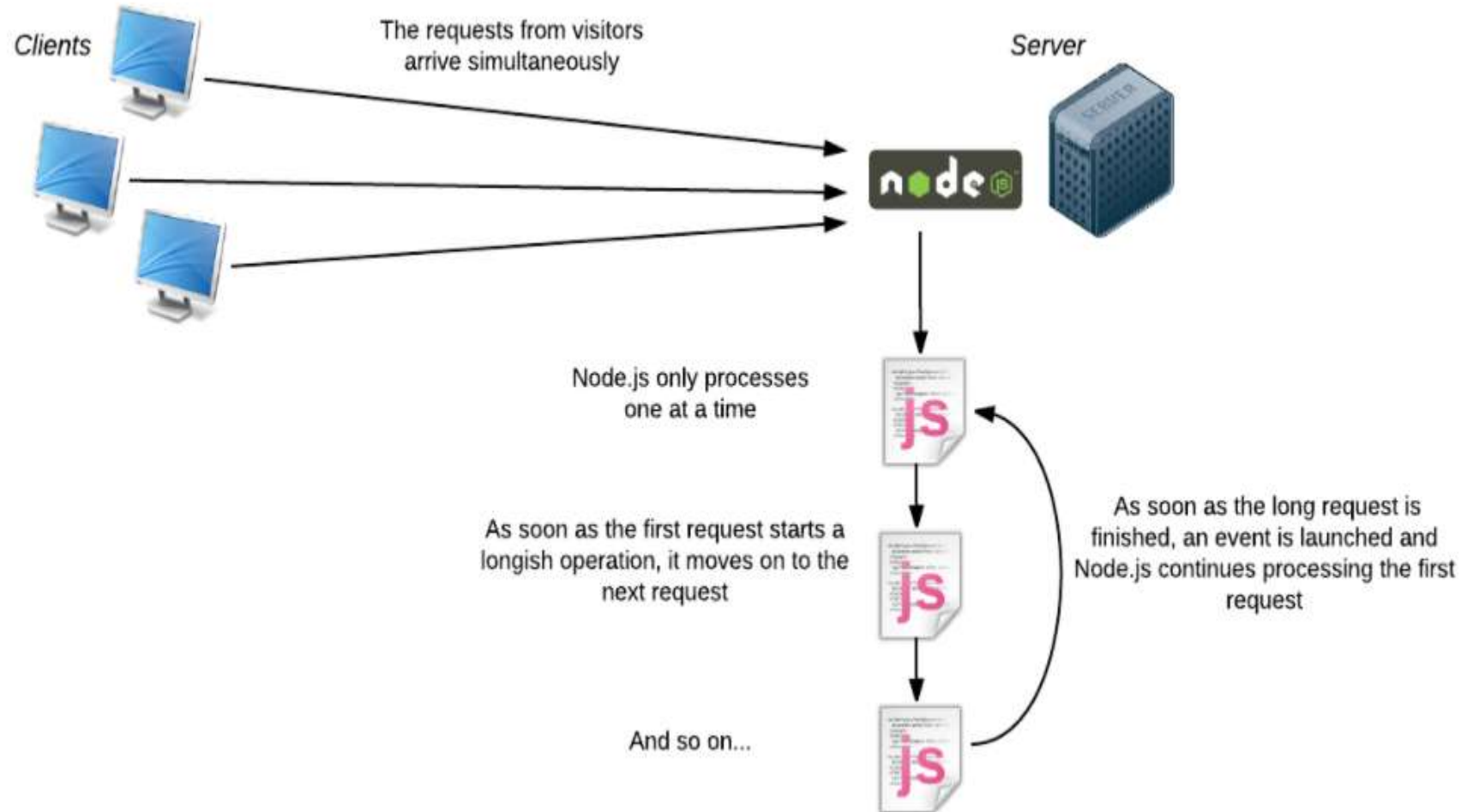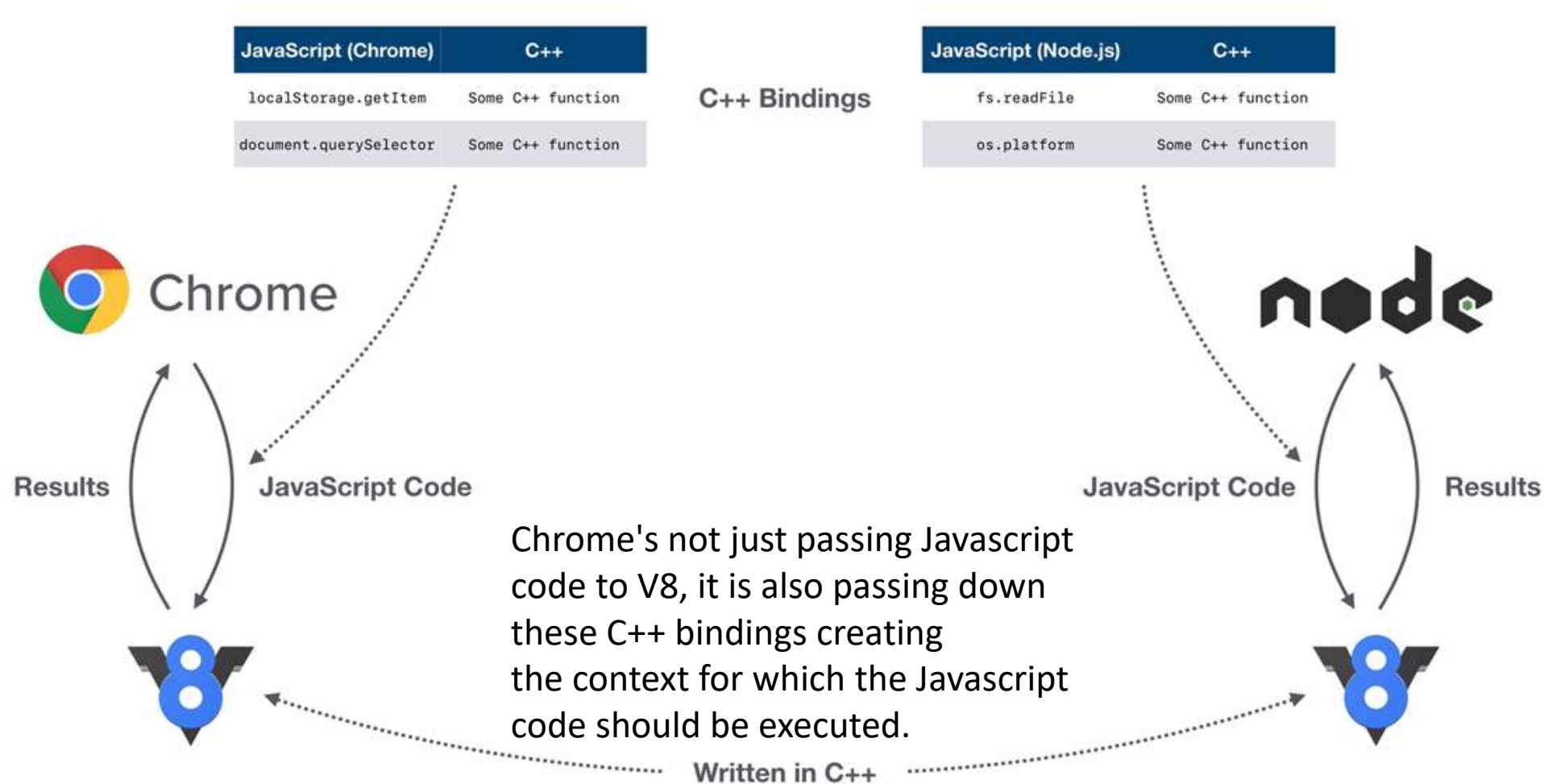
- Chrome needs to run some Javascript for a particular Web page, it doesn't run the JavaScript itself. It uses the V8 engine to get that done so it passes the code to V8 and it gets the results back. Same case with Node also to run a Javascript code.

- V8 is written in C++ . Chrome and Node are largely written in C++ because they both provide bindings when they're instantiating the V8 engine.

- This facilitates to create their own JavaScript runtime with interesting and novel features.

Example Instance:

Chrome to interact with the DOM when the DOM isn't part of JavaScript.

Node to interact with file system when the file system isn't part of JavaScript

The V8 JavaScript Engine

JavaScript (Chrome)	C++
localStorage.getItem	Some C++ function
document.querySelector	Some C++ function

C++ Bindings

JavaScript (Node.js)	C++
fs.readFile	Some C++ function
os.platform	Some C++ function

Chrome

Results   JavaScript Code

node

JavaScript Code   Results

Chrome's not just passing Javascript
code to V8, it is also passing down
these C++ bindings creating
the context for which the Javascript
code should be executed.

Written in C++

Features of Node.js

- Asynchronous and Event Driven

- Non Blocking I/O

- Very Fast

- Single Threaded but Highly Scalable

- No Buffering

- License

Features of Node.js

Very Fast

Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.

Fast Performance

Single Threaded but Highly Scalable

Uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers.

No Buffering

Node.js applications never buffer any data. These applications simply output the data in chunks.

Loading...

Asynchronous and Event Driven :
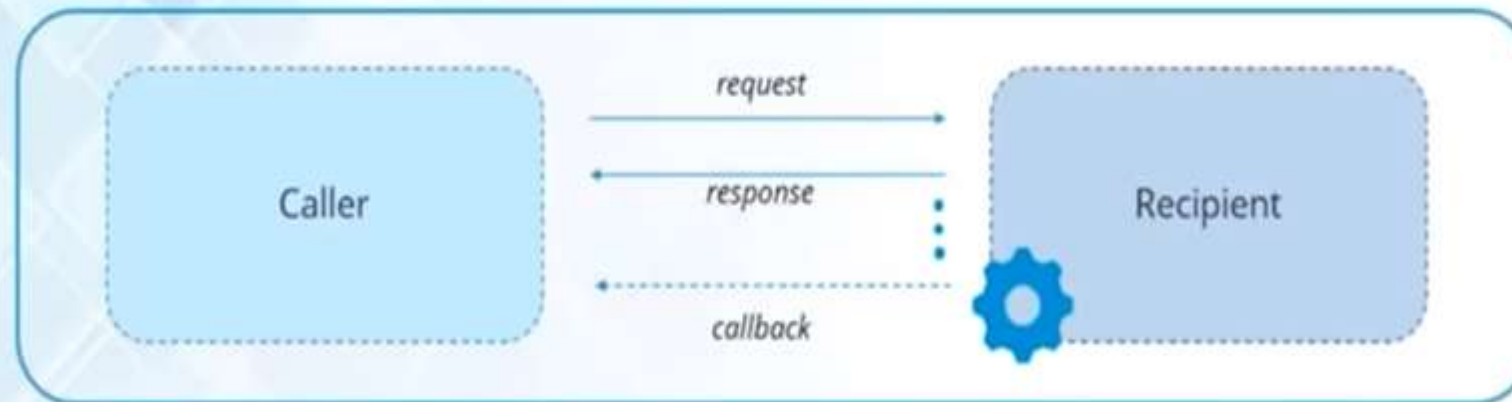
- When request is made to server, instead of waiting for the request to complete, server continues to process other requests
- When request processing completes, the response is sent to caller using callback mechanism

- Callback is an asynchronous equivalent for a function.

- A callback function is called at the completion of a given task. Node makes heavy use of callbacks.

- All the APIs of Node are written in such a way that they support callbacks.

For example, a function to read a file may start reading file and return the control to the execution environment immediately so that the next instruction can be executed. Once file I/O is complete, it will call the callback function while passing the callback function, the content of the file as a parameter. So there is no blocking or wait for File I/O. This makes Node.js highly scalable, as it can process a high number of requests without waiting for any function to return results.

NODE JS
Non- Blocking I/O

Blocking vs Non-Blocking

```
1 const getUserSync = require('./src/getUserSync')
2
3 const userOne = getUserSync(1)
4 console.log(userOne)
5
6 const userTwo = getUserSync(2)
7 console.log(userTwo)
8
9 const sum = 1 + 33
10 console.log(sum)
11
12
```

```
1 const getUser = require('./src/getUser')
2
3 getUser(1, (user) => {
4     console.log(user)
5 })
6
7 getUser(2, (user) => {
8     console.log(user)
9 })
10
11 const sum = 1 + 33
12 console.log(sum)
```

Fetching first user

Printing first user

Fetching second user

Printing second user

Calculate and print sum

Starting to fetch first user

Starting to fetch second user

Calculate and print sum

Printing first user

Printing second user

0 1 2 3 4 5 6 7 8   0 1 2 3 4 5 6 7 8

Blocking and Non- Blocking I/O

BLOCKING I/O

```javascript
var fs = require('fs');
var data = fs.readFileSync('test.txt');
console.log(data.toString());
console.log('End here');
```
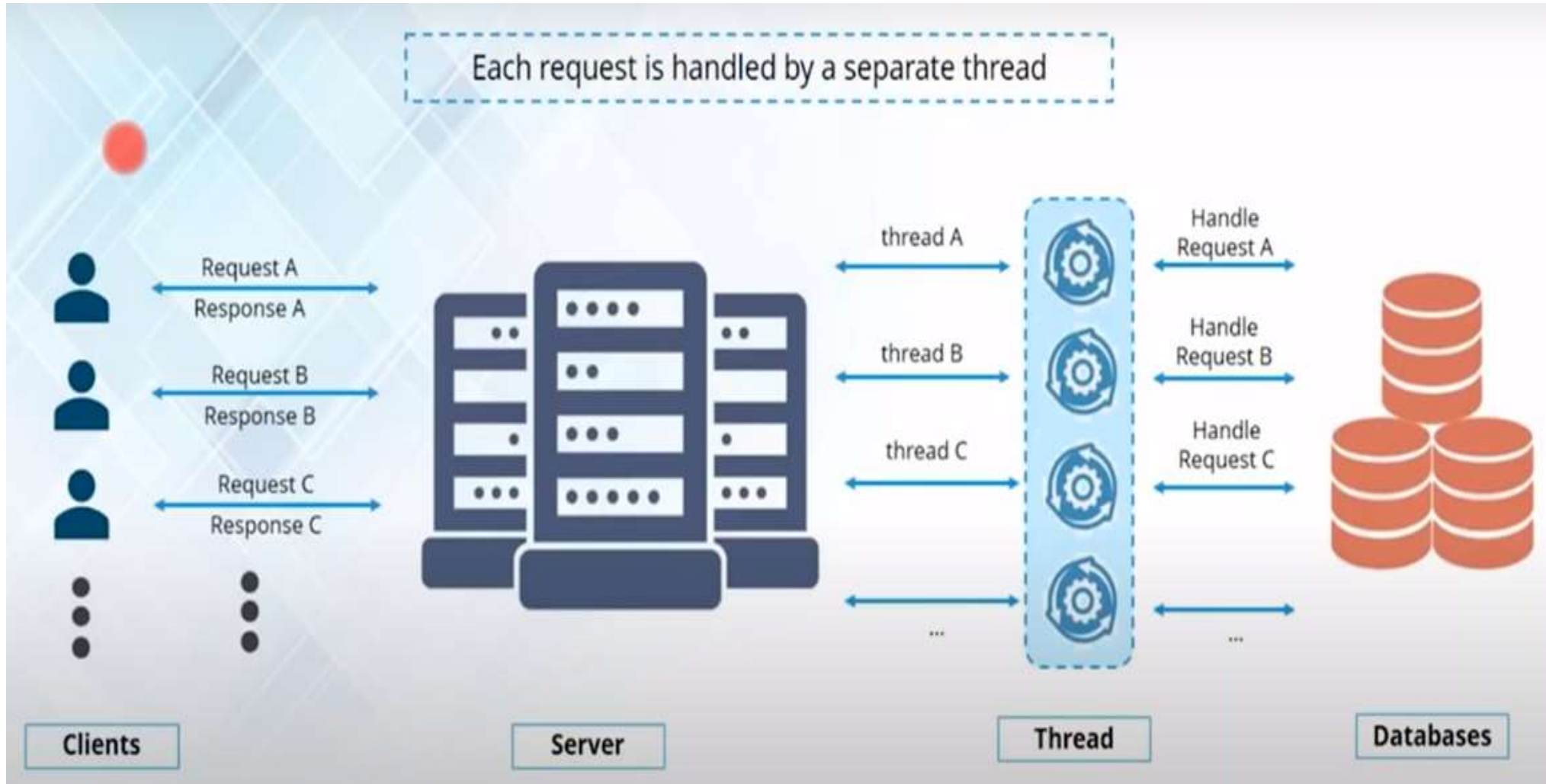
```javascript
var fs = require('fs');
fs.readFile('test.txt',function(err,data){
    if(err)
    {
        console.log(err);
    }
    setTimeout(()=>{
        console.log("PES University. Display after 2 seconds")
    },2000);
});
console.log('start here');
```
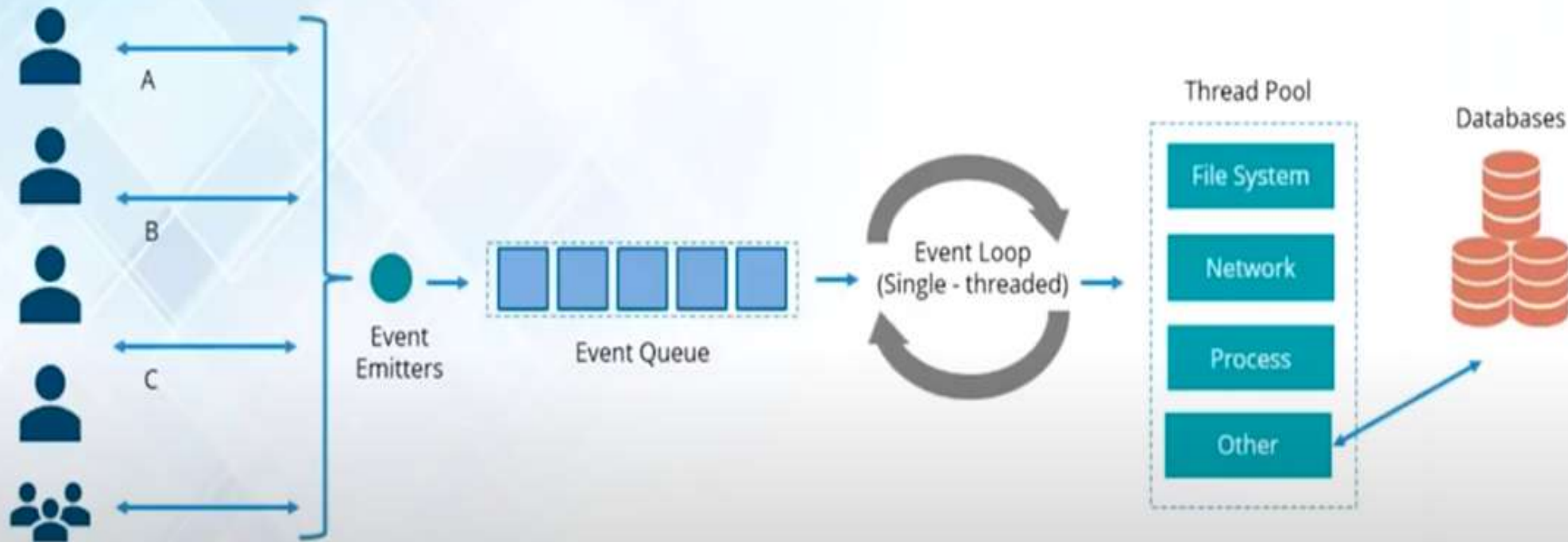
NON-BLOCKING I/O

Multi Threaded Model

- Node.js is event driven, handling all requests asynchronously from single thread
- Almost no function in Node directly performs I/O, so the process never blocks

NODE JS
Multi Threaded vs Asynchronous Event Driven Model

Multi-Threaded	Asynchronous Event-driven
Lock application / request with listener-workers threads	Only one thread, which repeatedly fetches an event
Using incoming-request model	Using queue and then processes it
Multithreaded server might block the request which might involve multiple events	Manually saves state and then goes on to process the next event
Using context switching	No contention and no context switches
Using multithreading environments where listener and workers threads are used frequently to take an incoming-request lock	Using asynchronous I/O facilities (callbacks, not poll/select or O_NONBLOCK) environments

- I/O bound Applications

- Data Streaming Applications

- Data Intensive Real-time Applications (DIRT)

- JSON APIs based Applications

- Single Page Applications

- Not for CPU intensive applications.

Nexflix used JavaScript and NodeJS to transform their website into a single page application.

NETFLIX

PayPal team developed the application simultaneously using Java and Javascript. The JavaScript team build the product both faster and more efficiently.

PayPal

Uber has built its massive driver / rider matching system on Node.js Distributed Web Architecture.

UBER

Node enables to build quality applications, deploy new features, write unit and integration tests easily.

Go Daddy.COM

When LinkedIn went to rebuild their Mobile application they used Node.js for their Mobile application server which acts as a REST endpoint for Mobile devices.

Linked in

They had two primary requirements: first to make the application as real time as possible. Second was to orchestrate a huge number of eBay-specific services.

ebay

- Customers interact with Uber applications and generating request to book a cab.

- Requests are sent to the Uber server to check in the geospatial databases for the cab.

- Server send back the card details or driver details back to the customer in form of response.

Disadvantage of Multi Threaded model in this case:

- Every request is assigned a new thread from thread pool and it gets exhausted. Scalability is very poor

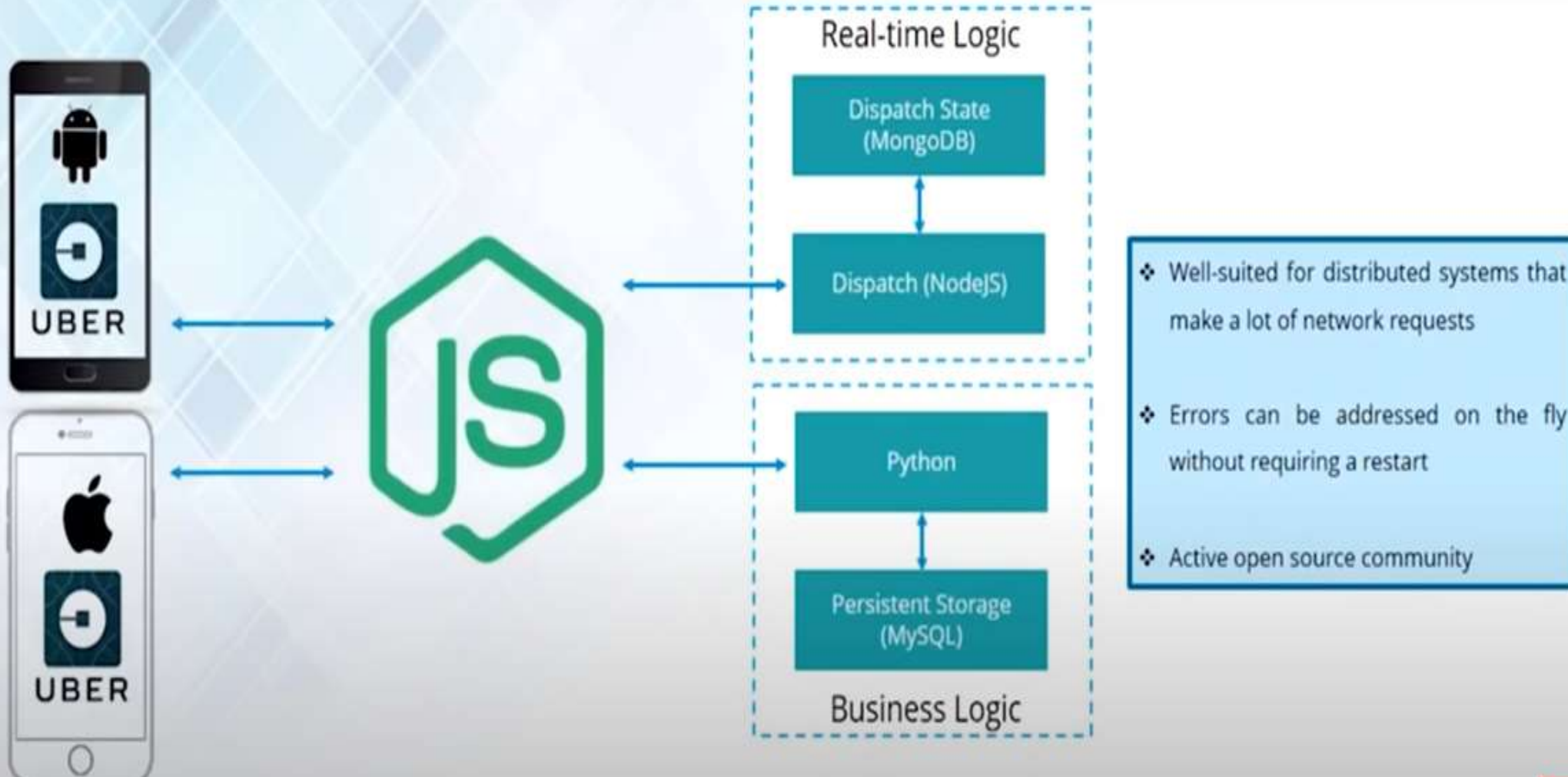- Whenever a thread is working on a shared resource. It acquires a lock on that resource.
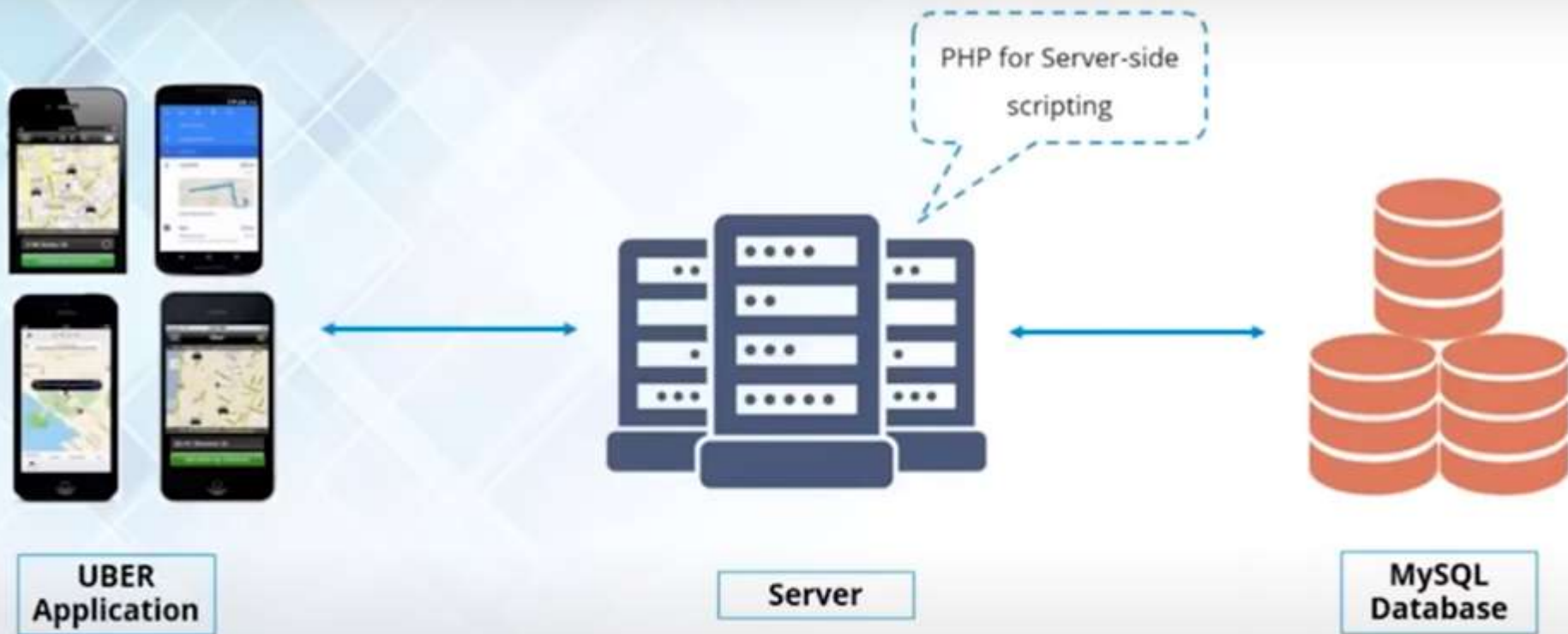
- Since PHP is a multithreaded language , each user's request is handled in a separate thread
- Reason was car dispatch operation was executed from multiple threads
- Once one car is dispatched for a user, in between the same car get dispatched to another user

- User performs an activity or event is generated, each new request is taken as an event.

- Event Emitter emit those events and then those events reside inside the event queue in the server.

- Events are executed using event Loops,which is a single thread mechanism

- A worker thread present inside the thread pool is assigned for each request.

- Only one thread in this event Loop will be handling the events directly and process will never get Blocked.

- Go to   https://nodejs.org/en/

- Look for the Latest Stable Version and download it

- After Installing, Verify the installation by giving the command as

```
C:\Users\DELL>node -v
v12.18.3
```

- Install a editor like Visual Code, Sublime Text or any suitable editors for running Node JS Applications

THANK YOU

Aruna S

Department of
Computer Science and Engineering

arunas@pes.edu