***“Rust Asynchronous Programming”***

**Background:**

Recently Rust Language adopted Asynchronous Programming and announced it in stable version. Asynchronous Programming is different from Normal programming. In normal programming only one task is done at a time which is easy for computer but in this way the system is under-utilized. So why one task is done at a time why not multiple tasks to be done at a time. Just like if we purchased a server so we need to utilize it well. That’s why we need to shift to concurrent programming so many tasks at a time can be done in an application.

**Introduction:**

Asynchronous programming is a concept which allows not blocking the program workflow when waiting for the results of certain actions. Here we’re talking about running code concurrently, or having multiple overlapping (in time) computations run on a single thread.

***Or***

Asynchronous Rust allows us to run multiple tasks concurrently on the same OS thread. Whatever the nature of your application a web server, a database or an operating system, using asynchronous programming you can get most out of the underlying hardware.

**Why Asynchronous Programming:**

Asynchronous programming lets us run multiple of these IO-bound computations at the same time on a single thread. In a typical threaded application multiple threads are created to perform multiple tasks which works fine but there are some limitations like overhead switching(one thread to another), data sharing or if a thread is not working also uses up valuable system resources or gaining additional resources. But asynchronous programming resolved all the problems and functions of multiple threads can be written using Rust’s async/await notation.

**Async/Await:**

The Rust Async ecosystem is still in progress and not final yet. The proposal here is to also use **async/await.** In Rust, **async fn** creates an asynchronous function which returns a **Future.**

The value returned by async fn is a Future. For anything to happen, the Future needs to be run on an executor. Unlike in synchronous method block on function is used to block whole thread. We can also use **.await** instead of **block\_on** inside **async fn. .await** doesn’t **block** the whole thread but wait for the specific **Future.**

**Block diagram:**

