Programação Assíncrona e Paralelismo

Techtalk TODO: Pesquisar AKKA

Johnathan Fercher

January 24, 2018

Table of contents

Introdução

Definições

Programação Assíncrona tem o propósito de possibilitar a execução de tarefas demoradas sem o bloqueio da execução.

Programação Paralela tem o propósito de possibilitar a execução de tarefas ao mesmo tempo.

Problema assíncrono

- Consumir uma API;
- Operações de CRUD em banco de dados;
- Quaisquer outras operações que demorem: **Treinamento de Inteligências Artificiais, Processamento de Imagens e Etc**;

Problema paralelo

- **Em um jogo:** Uma thread é responsável por obter os comandos do Joystick e outras *N* são responsáveis por controlar a renderização de objetos, comandos de adversários e etc;
- **Em um robô**: Uma thread é responsável pela leitura de sensores e outras *N* são responsáveis por controlar motores, realizar comunicação com outros robôs e etc;
- **Em um algoritmo de reconhecimento facial:** Pode-se dividir uma imagem em quadrantes, onde N threads serão responsáveis por aplicar filtros nas secções;

Programação Síncrona e Assíncrona em C#

Simples código síncrono

exemplo

Simples código assíncrono

exemplo

Transformando código síncronos em assíncronos

exemplo

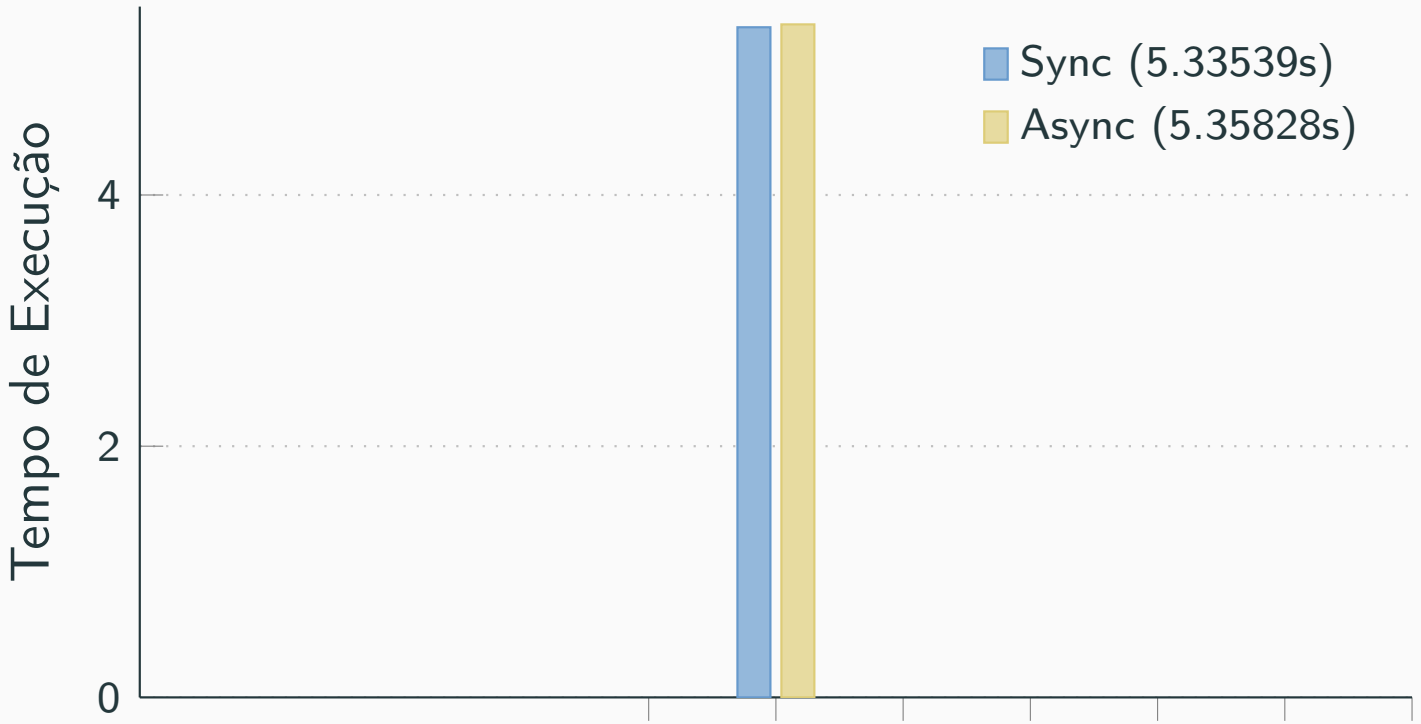
Transformando código assíncronos em síncronos

exemplo

Exemplos para Benchmark

Sync x Async

Benchmark



Fui tapeado?

Por que não houve ganhos de performance?

Programação Assíncrona apenas é responsável por não bloquear a execução.

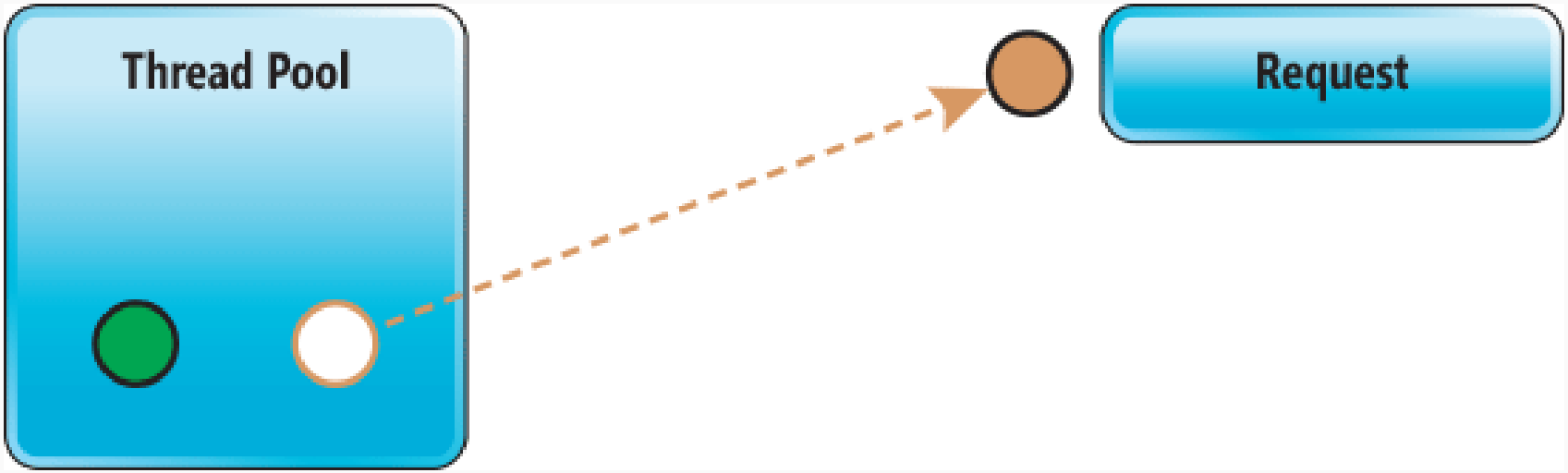
Então para que vou usar isso?

Depende da aplicação.

Vantagens da Programação Assíncrona

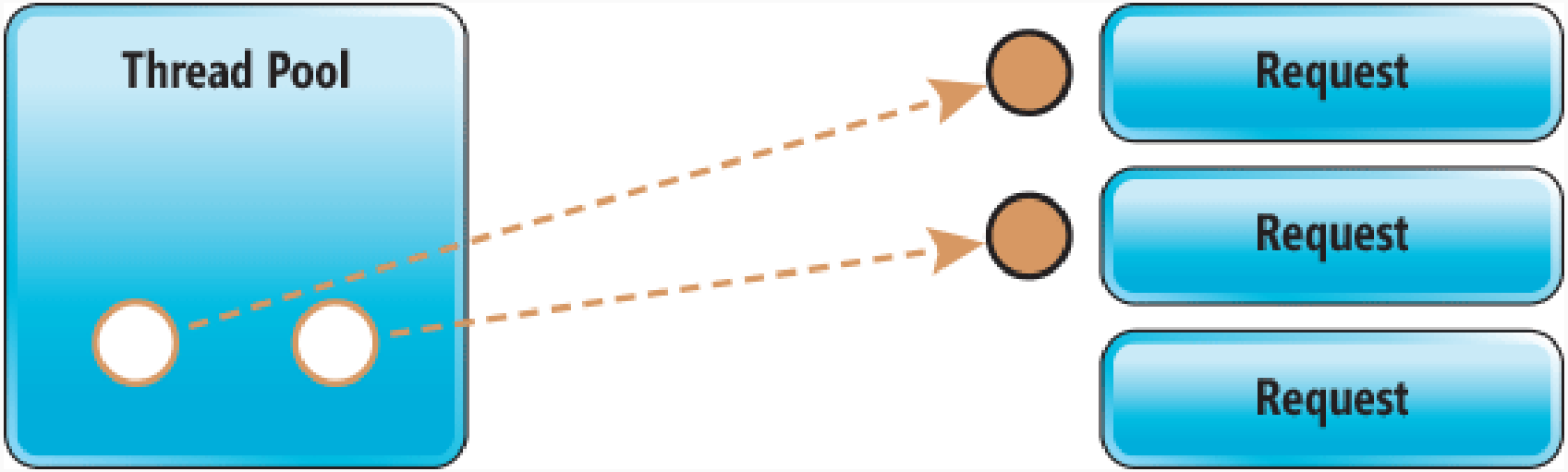
- **Em aplicações que possuem front-end**: A principal vantagem é o aprimoramento da interação com a aplicação;
- **Em um servidor**: A principal vantagem é o aprimoramento da escalabilidade;

Uma requisição síncrona no WebAPI



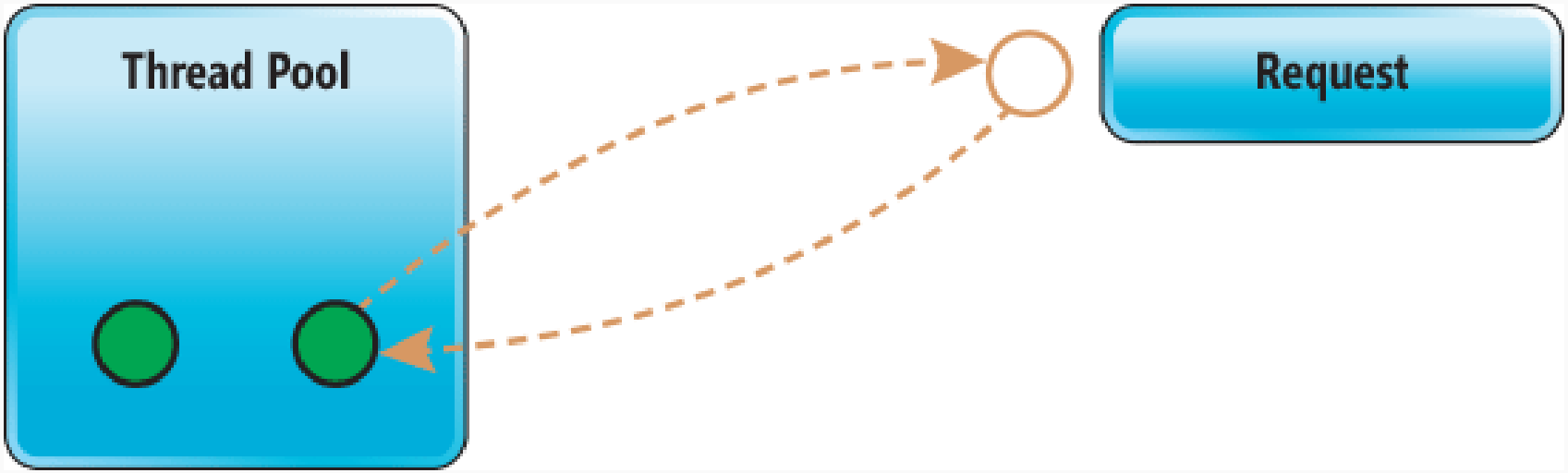
- Uma thread livre e outra thread bloqueada realizando **nada**;

Três requisições síncronas no WebAPI



- Duas threads bloqueadas realizando **nada** e uma requisição em espera;

Uma requisição assíncrona no WebAPI



- Thread realiza uma requisição bloqueante e retorna para a espera;

Ganhos

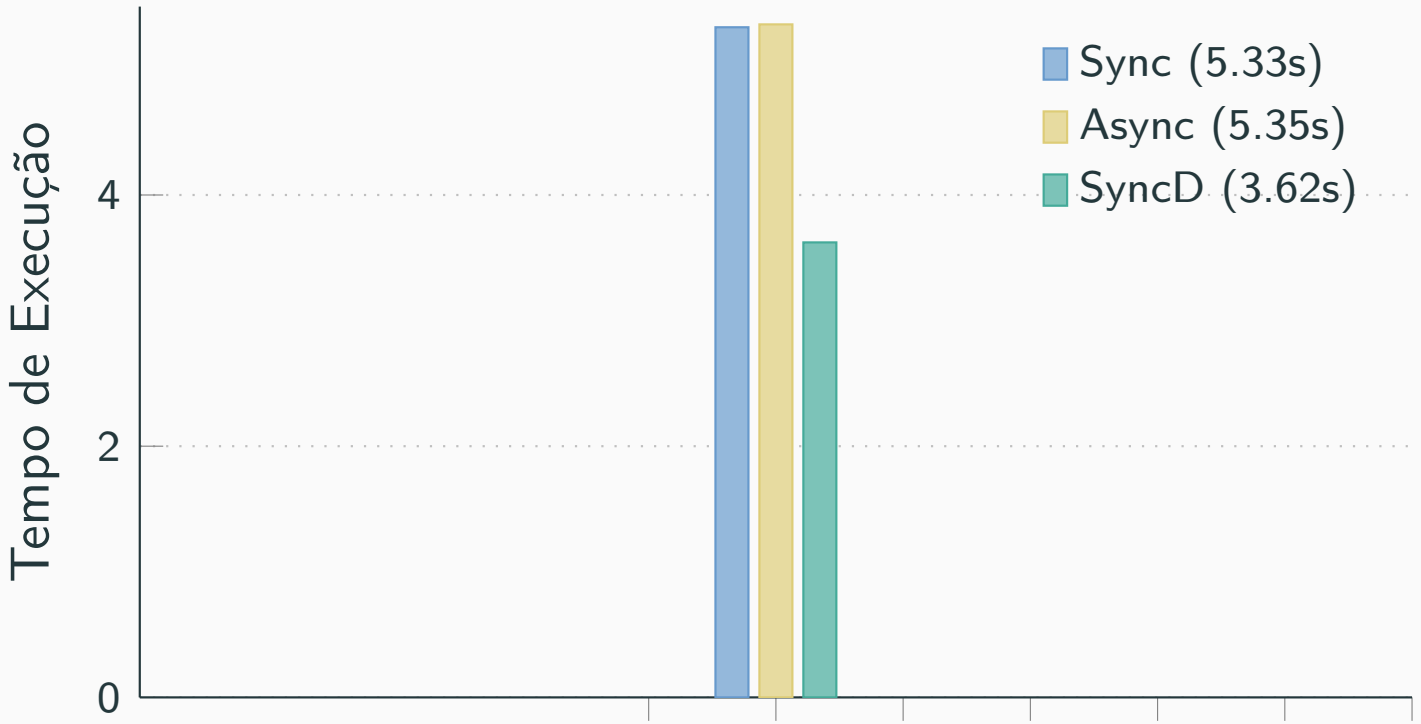
Redução de custos em relação à memória e redução do tempo de CPU desperdiçado;

Paralelismo em C#

Exemplo para Benchmark

SyncDataParallel

Benchmark



Perguntas Frequentes

Posso aplicar isso em qualquer problema?

Depende da quantidade de transformação de dados envolvida.

Quanto maior o número de threads melhor?

Existe um limiar onde a aplicação gasta mais tempo na comunicação entre threads do que o ganho de performance na execução paralela.

Tipos de Paralelismo

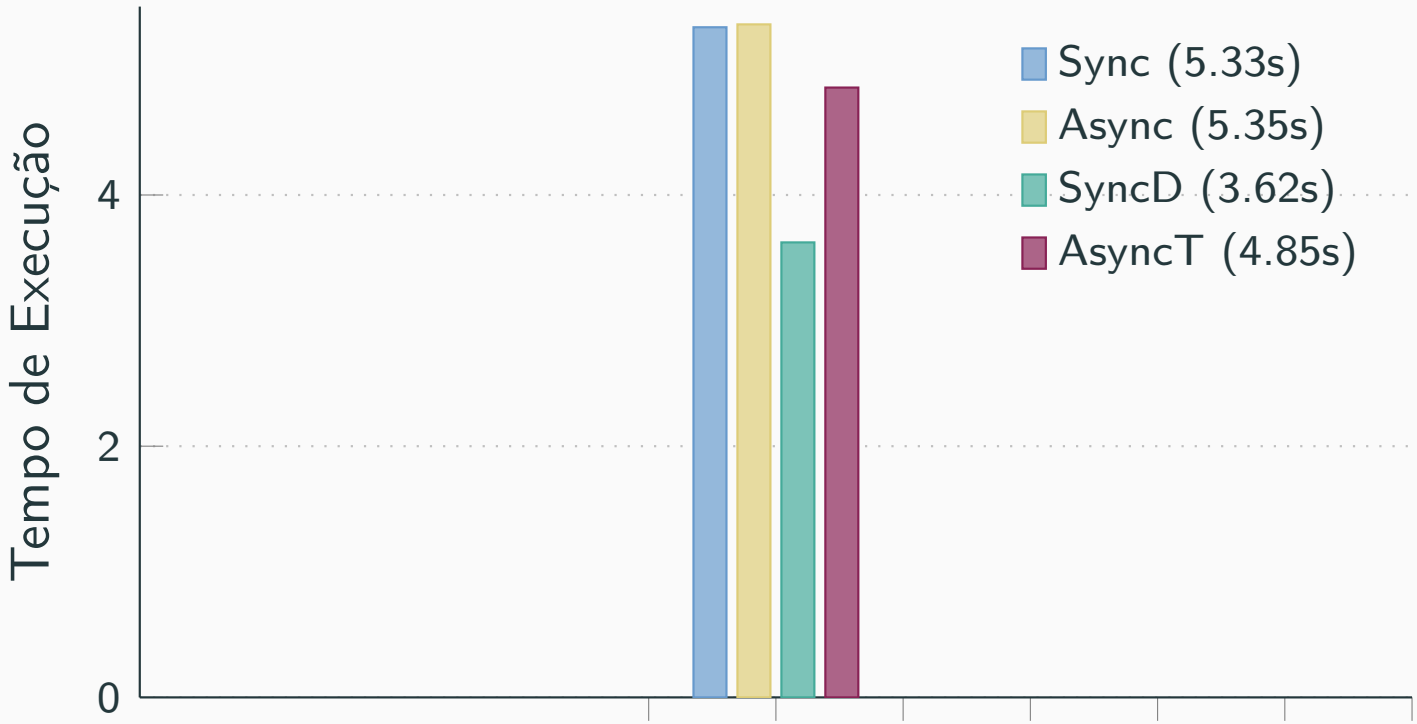
Paralelismo de Tarefas

Paralelismo de Dados

Exemplo para Benchmark

AsyncTaskParallel

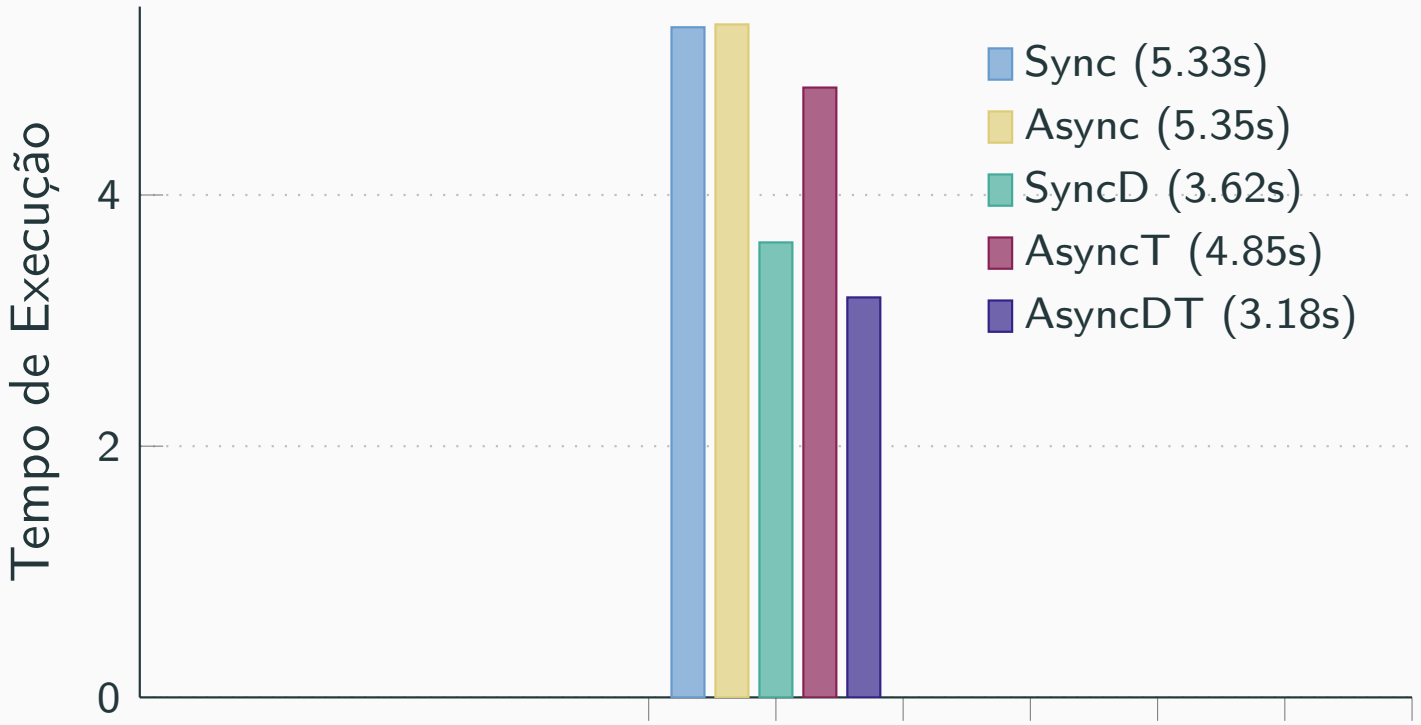
Benchmark


Tempo de Execução
0
2
4
Sync (5.33s)
Async (5.35s)
SyncD (3.62s)
AsyncT (4.85s)

Exemplo para Benchmark

AsyncTaskAndDataParallel

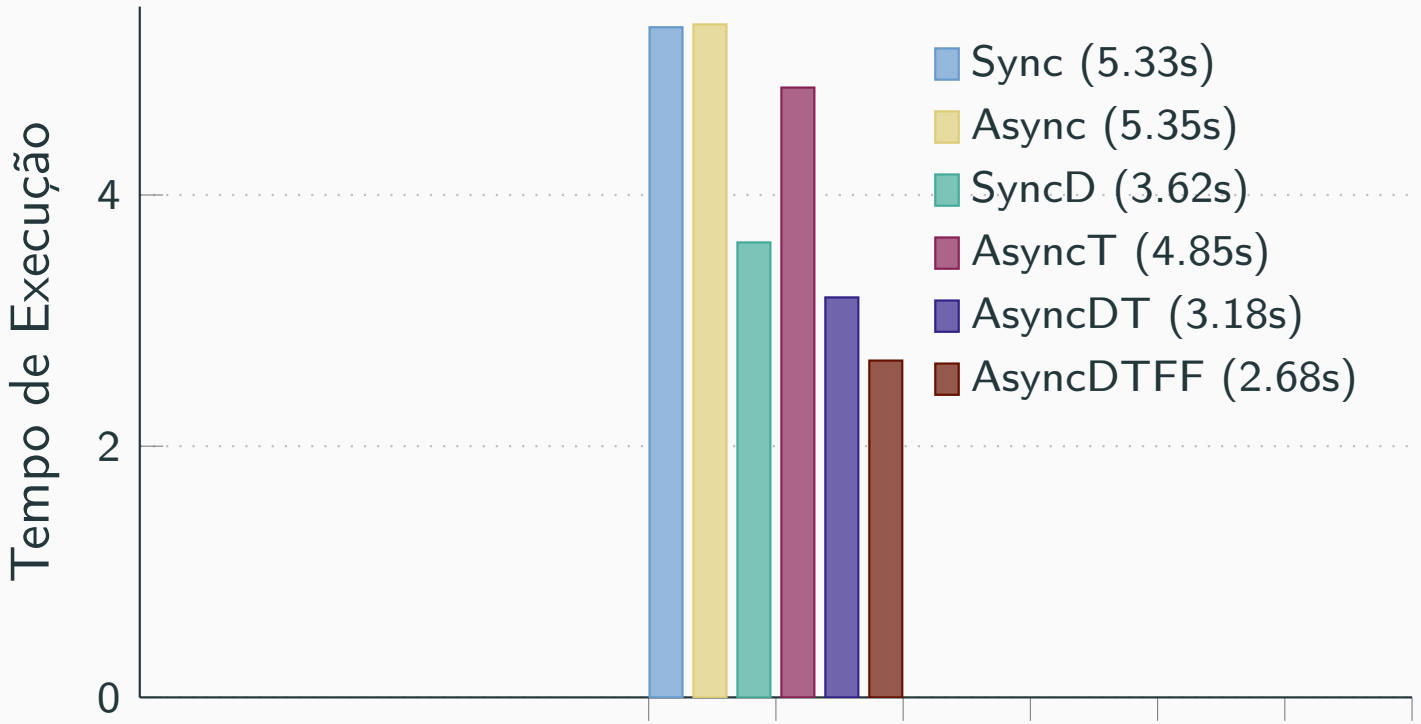
Benchmark


Tempo de Execução
0
2
4
Sync (5.33s)
Async (5.35s)
SyncD (3.62s)
AsyncT (4.85s)
AsyncDT (3.18s)

Exemplo para Benchmark

AsyncTaskAndDataParallel

Benchmark


Tempo de Execução
0
2
4
Sync (5.33s)
Async (5.35s)
SyncD (3.62s)
AsyncT (4.85s)
AsyncDT (3.18s)
AsyncDTFF (2.68s)

Problemas de Paralelismo

- Execução em ordem imprevisível;
- Condição de corrida;
- Execuções problemáticas;
- Falhas e Exceções;

Técnicas para evitar problemas

- Lock;
- Mutex;
- Semáforos;
- Monitor;
- Passagem de Mensagens;

Técnicas para evitar problemas

- Mestre e Escravo;
- Escritor e Leitores;
- 3 Monges;

Perguntas?