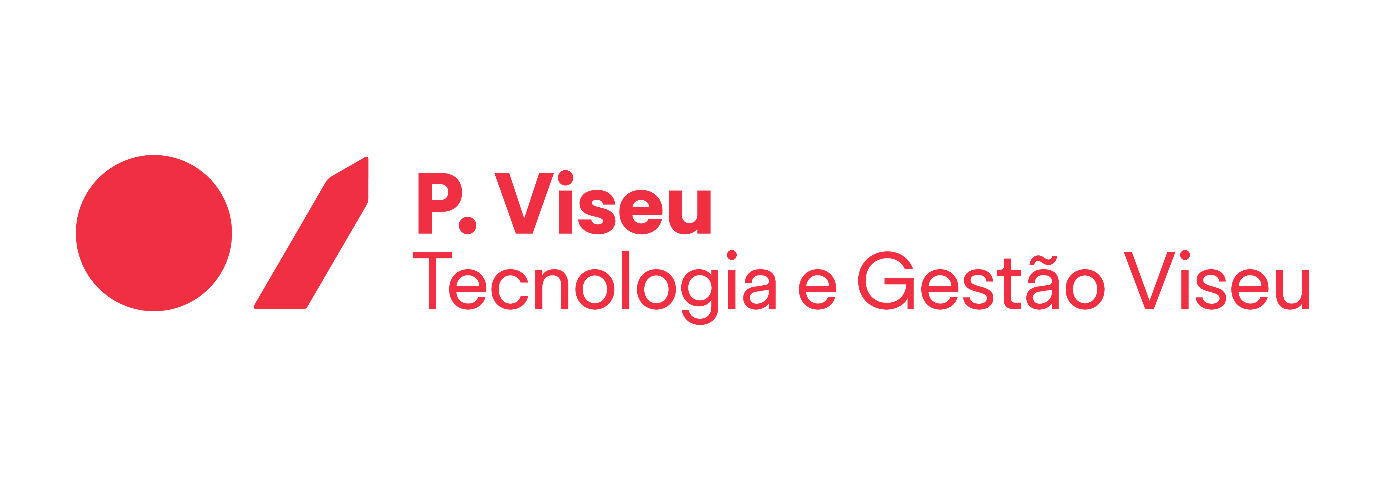
**IPV – Instituto Superior Politécnico de Viseu**

**ESTGV – Escola Superior de Tecnologia e Gestão de Viseu**

**Departamento de Informática**

****

**Análise da Solução Final**

**Sistemas Distribuídos**

**Realizado por**

**André Peixoto, nº 18738**

**Gabriel Raperger, nº 18509**

**Adelino Soares, nº 18737**

**Viseu, 2022**

**IPV – Instituto Superior Politécnico de Viseu**

**ESTGV – Escola Superior de Tecnologia e Gestão de Viseu**

**Departamento de Informática**

**Análise da Solução Final**

**Sistemas Distribuídos**

**Realizado por**

**André Peixoto, nº 18738**

**Gabriel Raperger, nº 18509**

**Adelino Soares, nº 18737**

**Viseu, 2022**

# Índice

[Índice V](#_Toc92661280)

[1 Introdução 1](#_Toc92661281)

[2 Descrição do projeto 3](#_Toc92661282)

[2.1 Funcionamento 3](#_Toc92661283)

[2.2 Pontos fracos 3](#_Toc92661284)

[3 Conclusão 5](#_Toc92661285)

# Introdução

Neste trabalho pretendemos desenvolver um sistema distribuído que permita distribuir pedidos de computação e rentabilizar a utilização de CPU pelos elementos do sistema, garantindo o equilíbrio da carga.

Para isso, criamos vários elementos com diferentes funções. Os carregadores funcionam como servidores de ficheiros permitindo armazenar os scripts a executar. Os processadores, por outro lado, são responsáveis por executar os scripts. Os estabilizadores são responsáveis por selecionar o processador que vai dar resposta ao pedido do cliente. Por último, os cérebros armazenam o resultado obtido da execução do script.

# Descrição do projeto

## Funcionamento

Primeiramente, o cliente começa por fazer upload do script no carregador através do protocolo SFTP, que é um método seguro de transferir ficheiros entre dois sistemas. Em seguida, o cliente envia um pedido de execução ao estabilizador que devolve o id do processador com mais recursos disponíveis. Após receber o id do processador, o cliente envia um pedido para esse processador, que em seguida vai fazer download do script armazenado no carregador e executá-lo. Por fim, o processador envia o resultado da execução para um dos cérebros que em seguida partilha o modelo com os restantes cérebros. Posteriormente o cliente solicita a um dos cérebros o modelo.

Foram ainda criados dois canais multicast, um deles para que os estabilizadores possam saber quais os processadores ativos e quais os seus recursos e tarefas. O outro para que os processadores possam descobrir os cérebros de forma descentralizada.

## Pontos fracos

Um dos pontos fracos do sistema era a não replicação dos modelos pelos restantes cérebros. Ou seja, quando um cliente solicitar o modelo ao cérebro, este poderia não o ter pois ele foi armazenado em outro. Para solucionar este problema criamos um método que permite partilhar o modelo recebido com os outros cérebros conferindo uma maior disponibilidade ao sistema. Semelhante ao modelo primary-backup de replicação, em que o processo primário comunica a alteração às replicas.

Um outro problema, era o facto de o cliente não conhecer os cérebros e fazer sempre o pedido ao mesmo cérebro. Isto pode resultar numa sobrecarga do cérebro. Para solucionar este problema a solução passaria por fazer com que os estabilizadores conheçam também todos os cérebros, ou seja, conectar os estabilizadores ao canal multicast e posteriormente, o cliente irá solicitar ao estabilizador um dos cérebros ao qual vai enviar um pedido.

Existe ainda um outro problema pois apesar de haver a replicação do modelo pelos cérebros, por qualquer motivo, um cérebro pode não possuir o modelo solicitado pelo cliente. Para solucionar este problema implementamos um método no cérebro para que este solicite o modelo aos outros cérebros quando não estiver em sua posse.

Por fim, existem ainda problemas em relação ao carregador uma vez que existe apenas um. Isto pode gerar uma sobrecarga do mesmo ou em caso de falha pode comprometer o sistema pois os scripts deixariam de estar disponíveis. A solução para este problema seria implementar replicas que em caso de falha, asseguram a resposta porem não implementamos esta solução.

# Conclusão

Por fim, apos implementar as soluções acima descritas, desenvolvemos um sistema com maior tolerância a falhas, uma vez que existem réplicas no caso de um processo parar de funcionar e com maior disponibilidade pois existem diversos processos para dar resposta aos pedidos dos clientes.

O repositório está no seguinte link <https://github.com/AndreFPeixoto/TrabalhoSDt.git>.