Computação na Nuvem (LEIRT e LEIC) – Versão 2019/2020

Síntese de atividades de laboratório

Laboratório nº: 1

Data: sexta-feira, 27 de março de 2020

Turma: 61D

Grupo: **Grupo 06**

Número e nome dos alunos presentes:

Número	*Nome*
43861	Francisco Chicharro
43874	Joao Florentino
Click or tap here to enter text.	Click or tap here to enter text.

1. Objetivo da atividade (descrição por palavras simples do que entendeu como objetivo da atividade);

Estabelecer uma comunicação entre servidor – cliente sendo o cliente o computador pessoal e o servidor uma máquina virtual da Google Cloud Platform (VM) onde esta é acedida através de uma sessão SSH.

2. Indicação das tecnologias e as ferramentas (*tools*) utilizadas;

-Google Cloud Platform para criação da VM;

-Bitvise SSH Client que permite o estabelecimento de uma sessão SSH para comunicação;

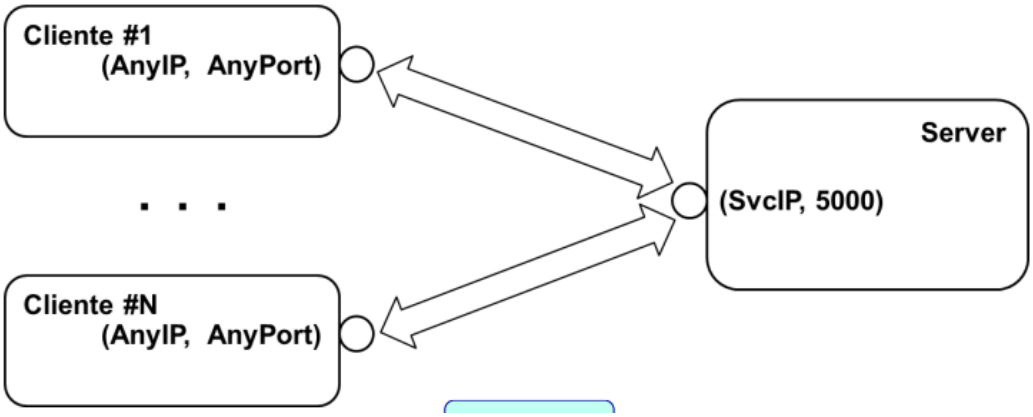
-Intelliij para executar as ações pretendidas através de código fornecido pelo Professor;

3. Descrição da arquitetura das partes (componentes) envolvidas, com eventuais diagramas:

Na primeira fase do trabalho, utilizou-se uma arquitetura cliente-servidor. Para validar a ligação, testou-se o código fornecido pelo docente, onde este pode funcionar de dois modos: Sequencial e concorrente

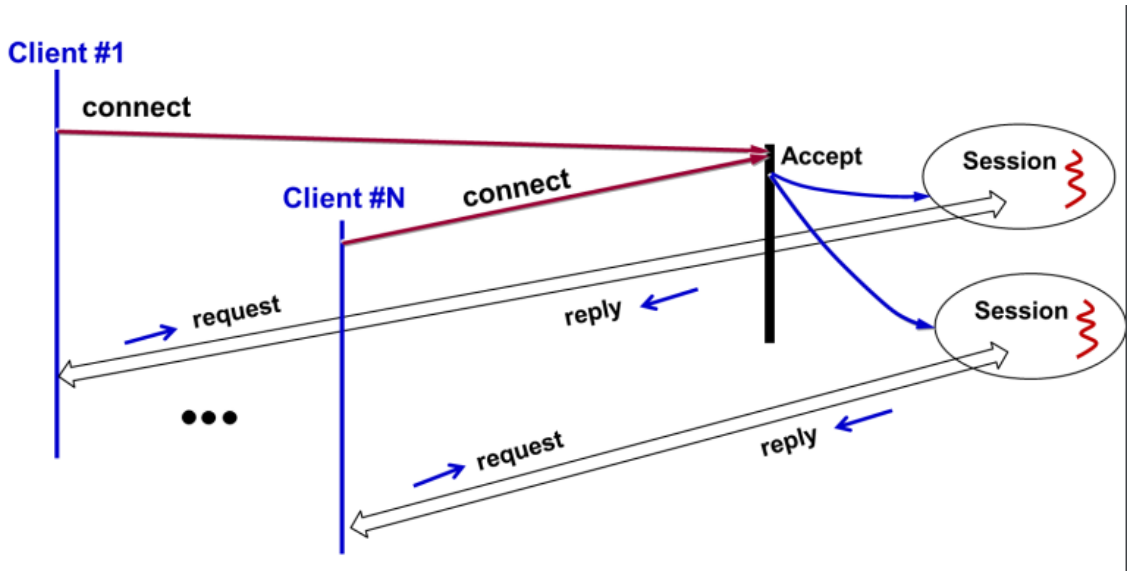
Modo sequencial: O servidor, após receber o pedido de um cliente, tem apenas uma tarefa (thread) a correr. Sendo o processamento feito em série, enquanto a ligação ao primeiro cliente estiver em execução ,não é possível ao servidor atender um segundo cliente,sendo necessário assim terminar a ligação com o primeiro para executar a tarefa do segundo cliente.

Neste modo, os clientes que estão em espera para ser atendidos, terão uma resposta ao seu pedido mais demorada, sendo que o tempo de latência será diferente para cada cliente.



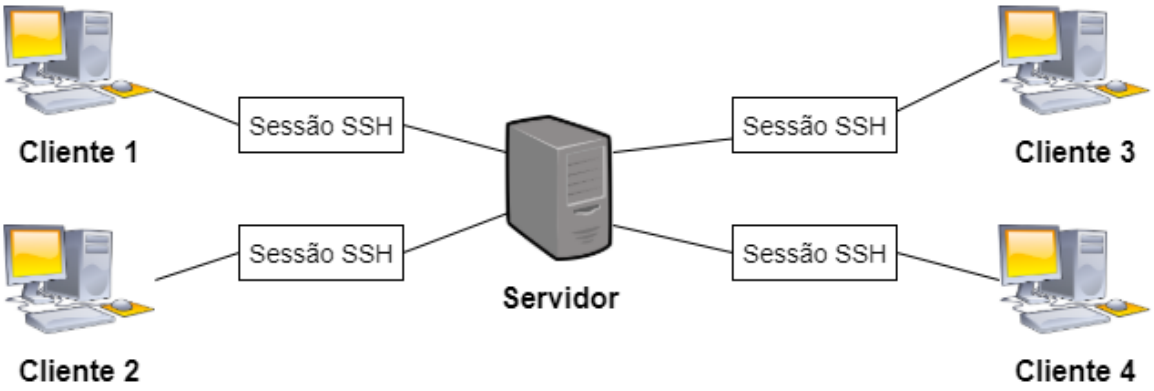
Modo concorrente: O servidor tem mais do que uma tarefa (thread) a correr ao mesmo tempo, permitindo agora a criação de sessões paralelas para múltiplos clientes distintos. Sendo agora o processamento de modo concorrente, não é necessário terminar a ligação com um cliente para atender outro, como visto anteriormente.

Este modo traz uma grande vantagem de reduzir a latência do atendimento ao cliente, uma vez que todos os atendimentos aos clientes terão tempo de processamento semelhantes.



Onde é visível que múltiplos clientes estão ligados ao servidor e este vai atender e executar todas as tarefas dos clientes distintos ao mesmo tempo.

Na segunda fase do trabalho, utilizou-se uma arquitetura cliente-servidor mas com uma sessão SSH no meio.O servidor utilizado anteriormente passou agora a ser uma VM criada na Google Cloud Platform, onde a sessão SSH permite a ligação entre o cliente e o novo servidor.

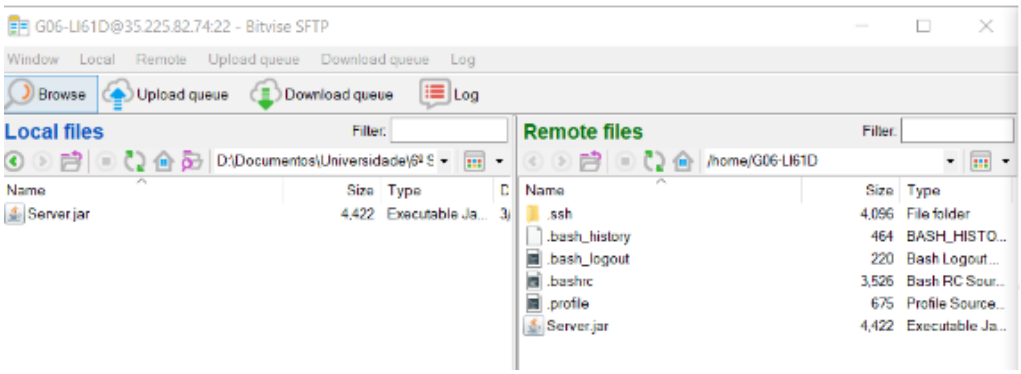


4. Resumo dos problemas encontrados e as soluções aplicadas:

   - No princípio importou-se a pasta total do código para o Intelliij como um projeto único que continha dois projetos distintos. Após realizar a ação build, conseguiu-se observar que o projeto não estava a criar a pasta artifacts que deveria conter o ficheiro Server.jar. Como tal, foi necessário abrir os projetos separadamente de forma a criar a pasta e o ficheiro solicitados.

   - Ao exportar a chave pública gerada no cliente SSH, reparou-se que não era possível que outro computador a importasse, como tal, procedeu-se á alteração do tipo de exportação de chave pública para chave privada, sendo assim possível alterar para o tipo de ficheiro pretendido

   -Foi necessário modificar a configuração do cliente, alterando os argumentos necessários para que este estabelecesse ligação com a máquina virtual(VM). No início utilizou-se como argumentos localhost e uma porta escolhida pelo grupo, que posteriormente foram alterados para IP externo da VM e para porta 80 que estava com estado “open”. Assim, foi possível fazer o upload ficheiro Server.jar necessário para os testes remotos:



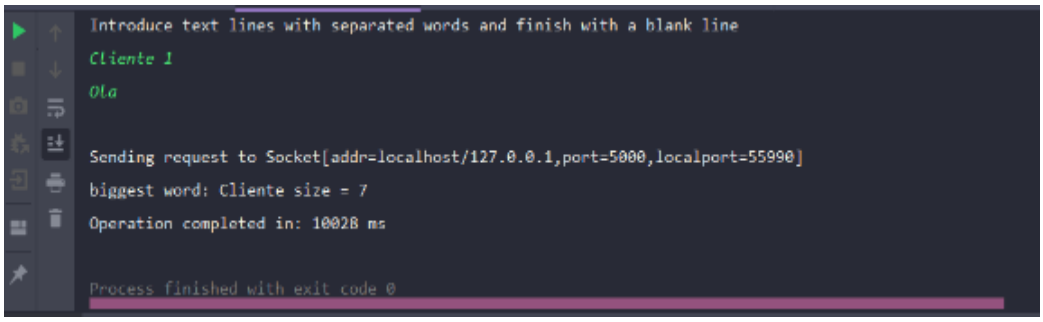
5. Indicação se a solução final é executável e demonstrável

   Após configurar tudo o que era necessário para estabelecer corretamente a ligação das arquiteturas apresentadas, realizou-se testes para comprovar o bom funcionamento das mesmas:

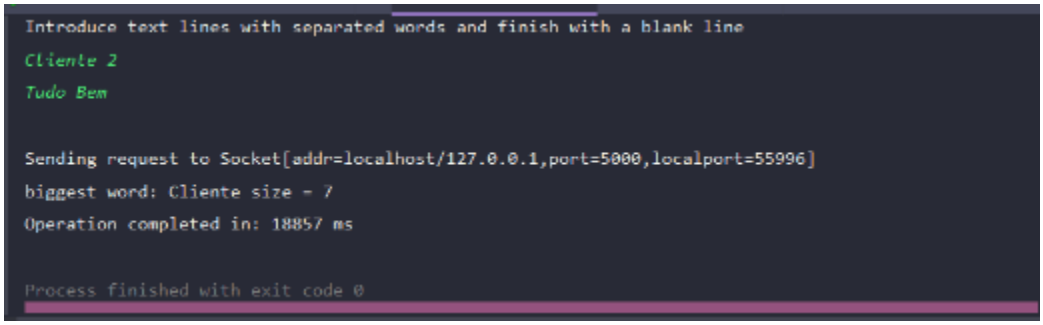
   Realização dos testes em modo sequencial: Neste teste foram utilizados três clientes (Cliente 1 e Cliente 2 e Cliente 3) onde se executaram os testes de ambas as arquiteturas.

Primeiro arquitetura:

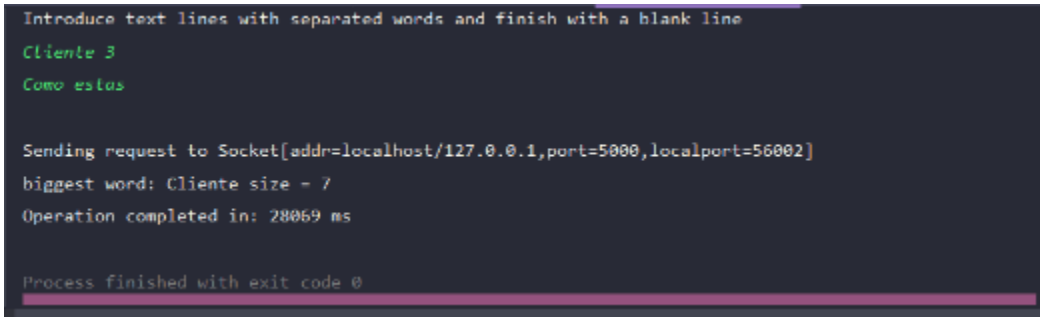
Cliente 1:



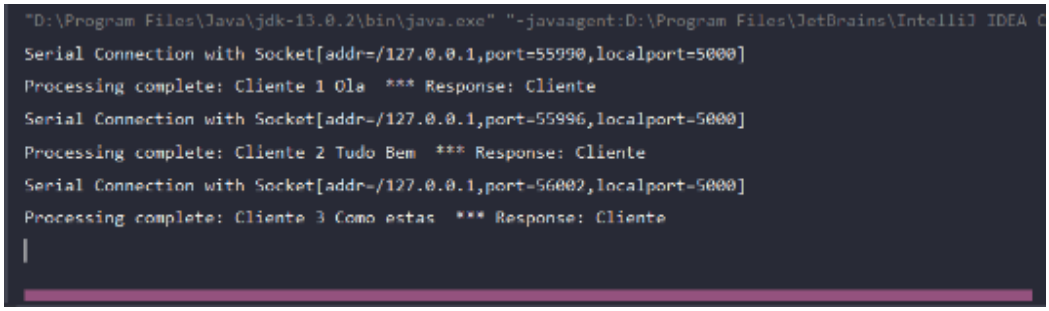
Cliente 2:



Cliente 3:



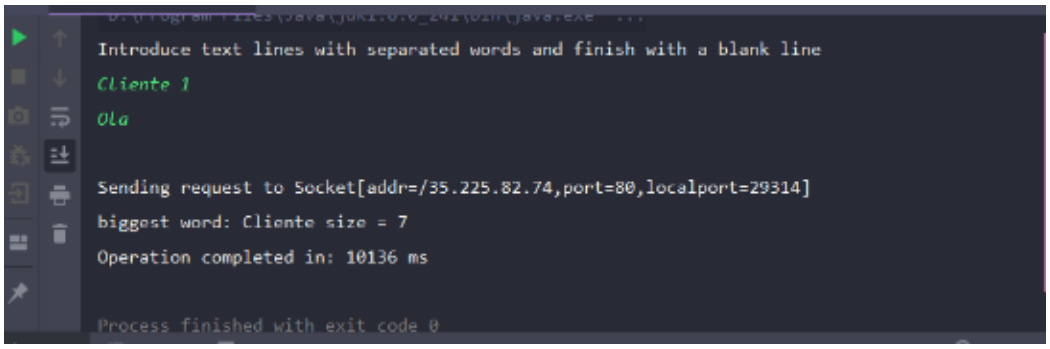
Onde o servidor recebeu corretamente os pedidos:



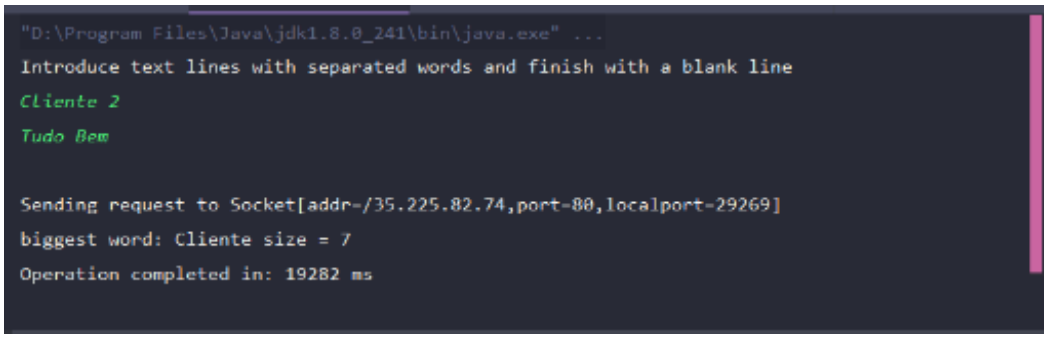
Comprova-se assim a teoria do tempo de processamento apresentado anteriormente, onde, em modo sequencial, os clientes iriam ter diferentes latências.

Realizou-se também os mesmos testes, utilizando agora a segunda arquitetura (após criada a máquina virtual):

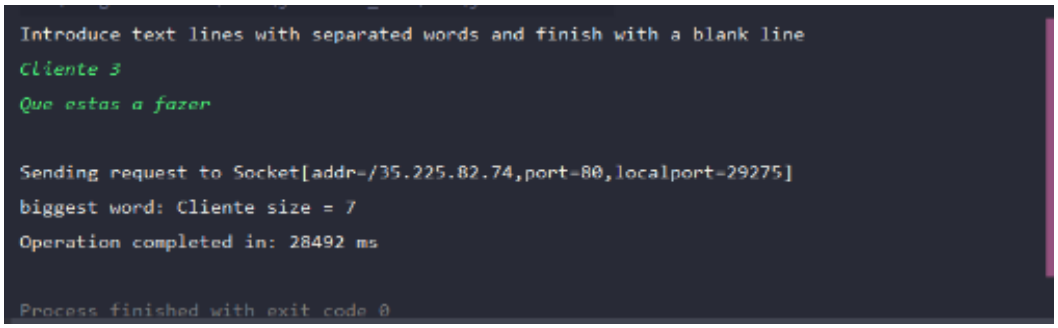
Cliente 1:



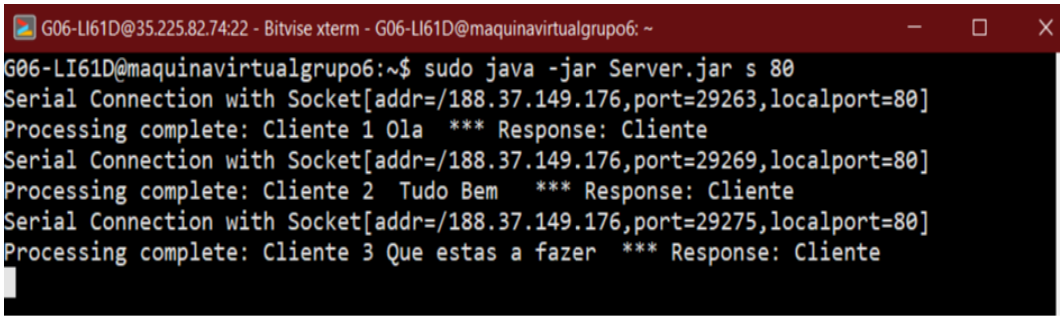
Cliente 2:



Cliente 3:



Nota-se que a máquina virtual recebeu corretamente os pedidos:

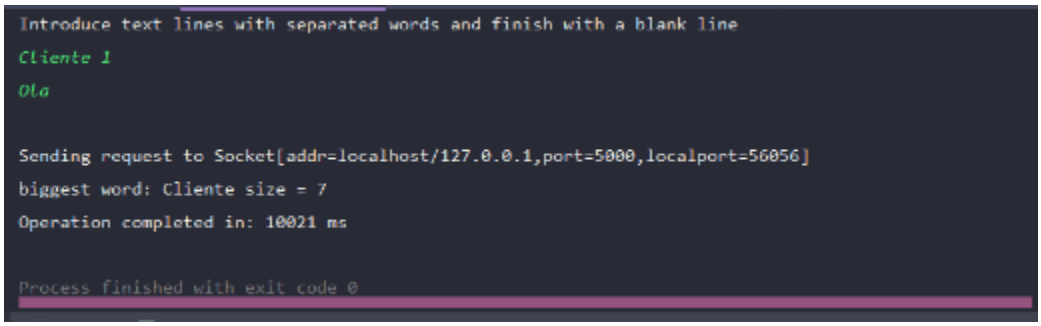


Onde os resultados dos tempos de latência são iguais aos obtidos na primeira arquitetura.

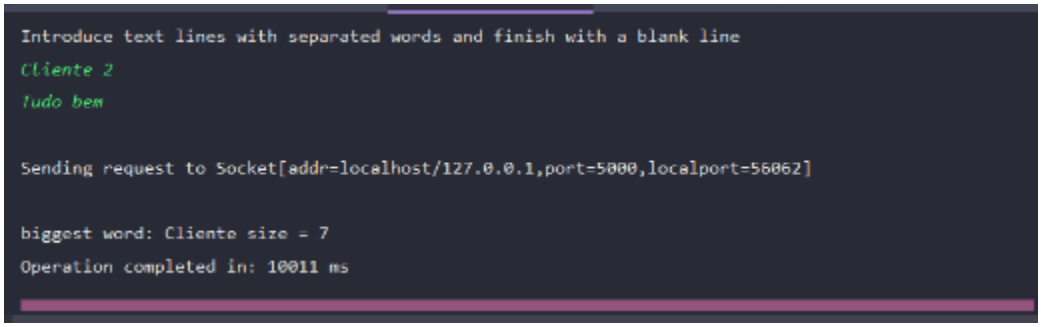
Realização dos testes concorrentes: Utilizando a mesma abordagem para o modo sequencional, utilizou-se novamente três clientes (Cliente 1, Cliente 2 e Cliente 3) onde se executaram os testes para as duas arquiteturas.

Primeiro arquitetura:

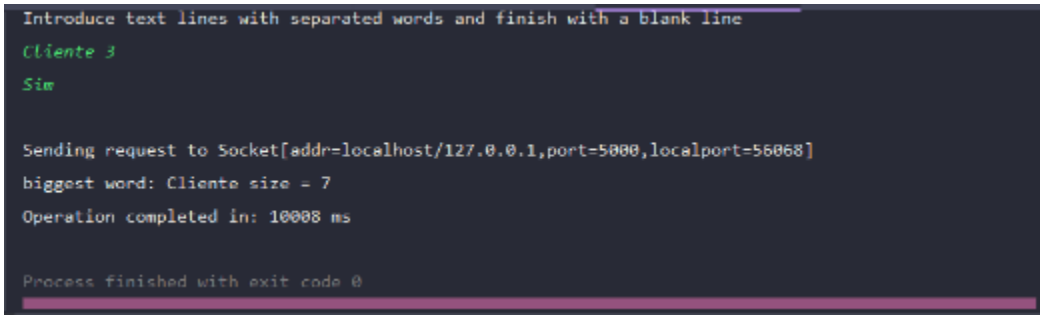
Cliente 1:



Cliente 2:



Cliente 3:



E para a segunda arquitetura:

Cliente 1:

```
Introduce text lines with separated words and finish with a blank line
Cliente 1
Ola

Sending request to Socket[addr=/35.225.82.74,port=80,localport=29314]
biggest word: Cliente size = 7
Operation completed in: 10136 ms

Process finished with exit code 0
```

Cliente 2:

```
Introduce text lines with separated words and finish with a blank line
Cliente 2
Tudo bem

Sending request to Socket[addr=/35.225.82.74,port=80,localport=29320]
biggest word: Cliente size = 7
Operation completed in: 10133 ms
```

Cliente 3:

```
Introduce text lines with separated words and finish with a blank line
Cliente 3
Que estas a fazer

Sending request to Socket[addr=/35.225.82.74,port=80,localport=29325]
biggest word: Cliente size = 7
Operation completed in: 10135 ms
```

Onde o servidor recebe corretamente os pedidos:

```
G06-LI61D@maquinavirtualgrupo6:~$ sudo java -jar Server.jar c 80
Server concurrent on port 80
Accepting new connections...
New connection with... Socket[addr=/188.37.149.176,port=29314,localport=80]
Accepting new connections...
New connection with... Socket[addr=/188.37.149.176,port=29320,localport=80]
Accepting new connections...
New connection with... Socket[addr=/188.37.149.176,port=29325,localport=80]
Accepting new connections...
Processing complete: Cliente 1  Ola  *** Response: Cliente
Session 1 terminates
Processing complete: Cliente 2  Tudo bem  *** Response: Cliente
Session 2 terminates
Processing complete: Cliente 3  Que estas a fazer  *** Response: Cliente
Session 3 terminates
```

Comprova-se novamente a teoria do tempo de processamento apresentado anteriormente, onde, em modo concorrente, os clientes iriam o mesmo tempo de processamento, sendo um processo mais vantajoso em tempo de espera

Cumpriu-se também objetivo de obter tempos de processamento semelhantes tanto para modo sequencial como para concorrente em ambas as arquiteturas, sendo assim mais vantajoso a utilização de uma máquina virtual em termos de poupança de recursos.

A solução apresentada é assim executável e demonstrável.

6. Conclusões e lições aprendidas

   Após conclusão do laboratório, o grupo conseguiu concluir que uma ligação em série, pelo facto dos pedidos dos clientes não serem executados em paralelo, o tempo de execução será mais elevado para os últimos clientes a efetuar pedidos. Pelo contrário, ao se utilizar uma ligação concorrente que suporta múltiplas, o tempo de execução será menor e praticamente igual para todos os clientes.

   Foi também possível aprender melhor a diferença entre uma ligação sequencial e uma ligação concorrente, assim como tirar proveito dos recursos da plataforma GCP que nos permitiu estabelecer uma ligação remota.

7. Auto-avaliação qualitativa por parte dos alunos

   Bom