



universidade de aveiro
theoria poiesis praxis

Universidade de Aveiro

Departamento de Electrónica, Telecomunicações e Informática

Arquiteturas de Software (2021/22)

Trabalho Prático III

Grupo n.º 22

João Carvalho, Nmec: 89059 - 50% de participação
João Pedro Pereira, Nmec: 106346 - 50% de participação

Performance

Um dos requisitos em relação à performance é que, a **computação**, equivalente ao número de iterações, deve ser uniformemente distribuída pelos servidores. Para isto se verificar, sempre que um *Load Balancer* recebe um pedido de um cliente, este envia uma mensagem para o *Monitor*, que contém a informação dos pedidos (e quantidade de iterações) em processamento de cada servidor, para ser actualizado acerca da ocupação de cada um, assim o *Load Balancer* saberá para qual servidor enviar o pedido.

Além disso, a implementação deve apresentar **concorrência** no processamento dos pedidos. Para ir de encontro a este facto, cada servidor apresenta três threads em execução para o processamento dos pedidos, além de uma queue de tamanho dois para conter pedidos em espera. Todas as threads de processamento estão à espera que a queue (servidor) receba pedidos, quando receber apenas um thread ficará responsável por processar o pedido e enviar uma resposta. Cada servidor apresenta um limite de 20 iterações máximas em processamento e espera. Quando se verifica a presença de dois pedidos em espera na queue a thread que acabou de processar o pedido actual e for buscar um novo, o pedido em espera com prioridade será o de menor *deadline*.

Availability

Caso um servidor ou *load balancer* crashar, os clientes não podem “perceber” e receber normalmente e corretamente as respostas que necessitam. Quem sabe que um destes parou de executar repentinamente é o *monitor*, usando mensagens de *heartbeats*, se parar de receber mensagens de um certo servidor ou *load balancer* declara que este crashou e procede às ações necessárias.

Um dos requisitos importantes no projeto quanto à disponibilidade é, a redistribuição dos pedidos de um servidor caso este crashar. Neste caso o *monitor* envia uma mensagem a notificar a paragem de um servidor com os pedidos que esse servidor estava a processar, depois o *load balancer* encarrega-se de redistribuí-los pelos restantes servidores ativos.

Se um *load balancer* crashar, existem duas possibilidades, se for o secundário, isto é, o *load balancer* que está inativo, toda a interação não sofrerá alterações. Porém se o *load balancer* primário crashar, o *monitor* (que sabe do ocorrido pelos *heartbeats*) tem de tomar as ações necessárias. O *monitor* declara que o *load balancer* secundário passa a ser primário e envia uma mensagem para este a notificá-lo. A partir desse momento, o novo *load balancer* primário começa a receber os novos pedidos do cliente.

Usability

Para ir de encontro aos requisitos para uma boa usabilidade do sistema as interfaces de cada entidade deve apresentar um uso fácil, intuitivo com possibilidade de rastreamento, verificação e validação.

A seguir encontram-se figuras com as interfaces de cada entidade.

Client: 3940

LB Host: LB Port:

Req ID	Nº Iterations	Deadline
3940000	4	40
3940001	4	40

Nº Requests: 2 Nº Replies: 2

Deadline: 40 sec

Number of Iterations:

Fig. 1: interface do cliente com separador dos pedidos

A interface do **cliente** apresenta dois separadores, uma com a informação de todos os pedidos feitos(acima), e outra com as respostas(abaixo). Podemos ainda definir as variáveis de cada pedido bem como os parâmetros de ligação ao *load balancer*.

Client: 20460

LB Host: LB Port:

Req ID	Nº Iterations	Deadline	Value
20460000	3	40	3,141
20460002	2	60	3,14
20460001	3	60	3,141

Nº Requests: 3 Nº Replies: 3

Deadline: 60 sec

Number of Iterations:

Fig. 2: interface do cliente com separador das respostas

Server: 14176

Monitor Host: Monitor Port:

LB Host: LB Port:

Req. ID	Nº Iterations	Deadline	Client ID	State
20460000	3	40	20460	DONE
20460001	3	60	20460	DONE
20460002	2	60	20460	DONE

Fig. 3: interface do servidor

A interface do **servidor** apresenta a informação relevante de cada pedido, sobretudo o estado de processamento: **Waiting**, se o pedido estiver em espera na queue; **<número de iteração atual>** se o pedido estiver a ser processado no momento; **DONE** se o pedido terminou de ser processado e **Rejected** se o pedido foi rejeitado.

Load Balancer: 19536

Monitor Host: Monitor Port:

LB Host: LB Port:

Req. ID	Nº Iterations	Client ID	Server ID
20460000	3	20460	14176
20460001	3	20460	14176
20460002	2	20460	14176

Fig. 4: interface do *load balancer*

A interface do **Load Balancer** apresenta a informação relevante de cada pedido, nomeadamente, o número de iterações, o cliente pelo qual o pedido foi feito e o server que lhe foi atribuído.

Monitor

Monitor Host: Monitor Port:

Servers | LBs | REQs

Server ID	State	Curr N° Iter	Total N° Iter
19176	UP	0	2

Fig. 5: interface do *monitor*, separados dos servidores

Monitor

Monitor Host: Monitor Port:

Servers | **LBs** | REQs

LB ID	State	Role
6108	UP	PRIMARY

Fig. 6: interface do *monitor*, separados dos load balancers

Monitor

Monitor Host: Monitor Port:

Servers | LBs | **REQs**

Req ID	Client ID	Server ID	N° Iter	Current Iter
20460000	20460	14176	3	Processed
20460001	20460	14176	3	Processed
20460002	20460	14176	2	Processed

Fig. 7: interface do *monitor*, separados dos pedidos

A interface do **monitor** apresenta três separadores: servidores, *load balancers*, e pedidos.

A respeito dos servidores, a interface apresenta o estado de execução de cada um, além dos pedidos que estão, no momento, na processar e o número total dos pedidos.

No caso dos *load balancers*, a interface apresenta o estado de execução e o papel que têm na interação, isto é, se está ativo ou não.

Por último, no separador dos pedidos, a interface mostra o cliente e servidor respetivos, o número de iterações do pedido e o seu estado atual de processamento.

Not work correctly

Terminado este trabalho, houve dificuldade num aspecto de *availability* nomeadamente relacionado com os *load balancers*. Não atingimos o objetivo de manter a interação das entidades a executar normalmente aquando de um crash no *load balancer* primário.