Interoperabilidade Semântica (FE02)

João Almeida (A75209), Luís Fernandes (A74748)

Março 2018

1 Introdução

A segunda fase deste trabalho, pretende estudar o desempenho do sistema criado na primeira fase, nomeadamente a sua capacidade no envio de mensagens entre as máquinas envolvidas no processo e registo das mesmas nas respetivas bases de dados de ambos os sistemas.

O desempenho é medido mediante o número de mensagens aleatórias, no formato tipico do HL7, que o sistema A consegue enviar ao sistema B, por unidade de tempo até que haja alguma inconsistência nesse envio. Pretende-se que o dito desempenho seja estudado em diferentes cenários para que as conclusões retiradas sejam o mais sólidas possível.

2 Implementação

Para proceder ao envio constante das mensagens do sistema A para o sistema B, foi implementado um programa na linguagem *Python*, chamado *sending.py*. Este programa foi idealizado de modo a não interferir na execução do resto do programa (inserção de pedidos, doentes, etc.). Estará totalmente dedicado a enviar as mensagens aleatórias para o sistema B. Por sua vez, a receção das mesmas, à semelhança da primeira fase, continuará a ser responsabilidade do programa *receiveB.py*. A comunicação entre os dois sistemas durante o envio e receção, será garantida (tal como na primeira fase) através da abertura de sockets num canal TCP/IP.

O programa *sending.py*, faz uso do módulo *threading*, existente na linguagem *Python*, para que a cada x segundos (definidos previamente), uma *thread* se encarregue de enviar, automaticamente, as ditas mensagens para o sistema B. As mensagens serão criadas recorrendo às mesmas funções que desempenhavam essa tarefa no programa da primeira fase.

A única alteração significativa no que diz respeito ao programa *receiveB.py*, trata-se do envio de um *ACK*, com um formato específico: ACK + ID pedido, para garantir algum controlo no fluxo das mensagens enviadas garantindo que as mesmas são rececionadas e registadas na base de dados do sistema B, evitando que se percam um conjunto de mensagens no processo de comunicação das mesmas. É responsabilidade do sistema A, nomeadamente de *sending.py*, continuar

o envio constante de mensagens, no intervalo definido, interrompendo excepcionalmente esse envio, cada vez que o *ACK* referente a uma mensagem, não seja recebido e procedendo ao reenvio da mesma x segundos após a ocorrência, caso este cenário persista.

O programa *receiveB.py* sofreu alterações que permitissem o envio de uma mensagem *ACK X*, ao sistema A, informando o mesmo da correta receção e registo da mensagem X, e que permitissem a contabilização das mesmas.

O programa *sending.py* está implementado de tal forma, que basta executá-lo uma vez num qualquer terminal, para que o envio seja iniciado, isto claro, caso o sistema B esteja preparado para a receção (*receiveB.py* em execução).

3 Medidas de Desempenho

As medidas de desempenho que decidimos considerar, consistem na variação dos intervalos de tempo entre os quais as mensagens serão enviadas ao sistema B e consequentemente no número de mensagens que serão enviadas por unidade de tempo, que irá ser naturalmente maior, quanto menor o intervalo de envio definido.

Enviadas/minuto	Recebidas/minuto	Intervalo (s)	*ACK*	Falha na receção
60	60	1.0	✓	Envio e receção normal
180	180	0.3	✓	Cria fila de espera
600	5	0.1	✓	Cria fila de espera e encerra envio inesperad
300	0	0.2	✓	Encerra envio inesperadamente

4 Melhorias e Trabalho Futuro

Por não termos experiência e conhecimento na área, não conseguimos classificar de maneira segura qual a qualidade da performance conseguida pelo nosso programa, por não sabermos quais são os valores de referência nesta situação. Essencialmente, e resumindo os dados acima recolhidos, no melhor caso, o nosso programa consegue enviar 180 mensagens por minuto (intervalo de envio de 0.3s), com a devida receção do ACK e registo nas bases de dados de ambos os sistemas, totalizando 259200 mensagens por dia.

Entendemos ainda que uma possível melhoria ao nosso programa, passaria pela alteração do modo como as mensagens são registadas nas bases de dados do sistema A. Neste momento, a nossa implementação prevê o registo das mesmas na tabela Pedido e na tabela Worklist. Visto que a última referencia os pedidos, na forma dos seus identificadores numéricos, torna-se um pouco redundante guardar a mesma informação em duas frentes diferentes. A remoção desta operação de escrita, levaria (nas nossas previsões) a uma melhoria na capacidade de envio de mensagens que o sistema A apresenta.

Admitimos ainda, que devido à nossa inexperiência na linguagem de programação Python, as estruturas de dados usadas, assim como as implementações

de algumas funções que afetam aquela que vai ser a performance final do programa, podem não ter sido as mais corretas e podem ser sujeitas a otimizações no sentido de as melhorar.

Na domínio do envio pelo canal TCP/IP, a única otimização que poderia eventualmente ser considerada para o futuro seria a utilização de agregação de mensagens, para usufruir da capacidade total em termos de tamanho, que o canal dispõe. Neste momento, cada mensagem ocupa, em média 300 bytes, ou seja, aproximadamente 1/3 da capacidade total que é possível enviar de cada vez por um socket num canal TCP/IP. Existe portanto, espaço para otimizar este envio, caso consideremos juntar 3 mensagens num só envio, aumentando significativamente deste modo, o número total de mensagens rececionadas, mantendo o mesmo número de envios por unidade de tempo e usufruindo da capacidade total do canal.

Por último, será no nosso entender importante também referir, que a própria capacidade das máquinas (PCs) envolvidas no processo, influência a performance final do nosso programa.