Universidade do Minho

Mestrado em Engenharia Informática

Administração de Base de Dados

Ano lectivo 2014/2015

Trabalho Prático

Relatório Final

Autores:
Luís Miguel Silva (pg)
Luís Miguel Pinto (pg27756)
Pedro Carneiro (pg25324)

Braga, 21 de Dezembro de 2014

Conteúdo

1 Introdução

2 Resultados de Desempenho na configuração híbrida *TPC-C + CH-benCHmark*

2.1 *Serializable* como Método de Isolamento

# clientes	# pedidos	pedidos/s	lat. média (s)	lat. perct. 99 (s)
2	5068	84.4667	1.0291	1.0702
4	3226	53.7653	1.1079	2.0538
8	1377	22.9500	1.3489	14.7304
16	703	11.7166	1.8898	24.8298

Tabela 1: Resultados obtidos para dois (2) armazéns

# clientes	# pedidos	pedidos/s	lat. média (s)	lat. perct. 99 (s)
2	2503	41.7166	1.0631	1.0909
4	434	7.2333	0.9899	1.2105
8	516	8.5999	2.1197	57.2194
16	474	7.9000	1.6451	25.3641

Tabela 2: Resultados obtidos para quatro (4) armazéns

2.2 *Repeatable Read* como Método de Isolamento

# clientes	# pedidos	pedidos/s	lat. média (s)	lat. perct. 99 (s)
2	17138	285.6333	1.0078	1.0305
4	18435	307.2458	1.0196	1.1275
8	15426	257.0989	1.0439	1.2903
16	13315	221.9153	1.0917	1.6116

Tabela 3: Resultados obtidos para dois (2) armazéns

# clientes	# pedidos	pedidos/s	lat. média (s)	lat. perct. 99 (s)
2	16818	280.2991	1.0088	1.0297
4	20372	339.5289	1.0113	1.1042
8	17256	287.5948	1.0304	1.2829
16	8744	145.7319	1.1217	2.4365

Tabela 4: Resultados obtidos para quatro (4) armazéns

2.3 *Read Uncommitted* como Método de Isolamento

# clientes	# pedidos	pedidos/s	lat. média (s)	lat. perct. 99 (s)
2	17797	296.6167	1.0090	1.0291
4	20594	343.2303	1.0171	1.1176
8	20852	347.5300	1.0278	1.2382
16	17933	298.8668	1.0770	1.4699

Tabela 5: Resultados obtidos para dois (2) armazéns

# clientes	# pedidos	pedidos/s	lat. média (s)	lat. perct. 99 (s)
2	17253	287.5477	1.0080	1.0308
4	18406	306.7542	1.0194	1.1353
8	16434	273.8942	1.0306	1.2895
16	18418	306.9657	1.0667	1.5075

Tabela 6: Resultados obtidos para quatro (4) armazéns

2.4 *Read Committed* como Método de Isolamento

# clientes	# pedidos	pedidos/s	lat. média (s)	lat. perct. 99 (s)
2	14810	246.8311	1.0114	1.0446
4	20792	346.5310	1.0170	1.1070
8	21509	358.4819	1.0310	1.2144
16	12534	208.8956	1.0901	1.9881

Tabela 7: Resultados obtidos para dois (2) armazéns

# clientes	# pedidos	pedidos/s	lat. média (s)	lat. perct. 99 (s)
2	17495	291.5808	1.0051	1.0264
4	19219	320.3136	1.0165	1.1110
8	18722	312.0306	1.0276	1.2623
16	18302	305.0299	1.0609	1.5071

Tabela 8: Resultados obtidos para quatro (4) armazéns

2.5 Comparação dos Métodos de Isolamento e Conclusões Finais

3 Otimização e/ou justificação do desempenho tendo em conta as *queries* analisadas

4 Otimização e/ou justificação do desempenho tendo em conta os parâmetros de configuração do PostgreSQL

4.1 Configuração do *shared_buffers*

O parâmetro de configuração *shared_buffers* determina a quantidade de memória RAM dedicada ao *PostgreSQL* que vai ser usada para armazenar dados enquanto executa queries.

Inicialmente, podia-se prever que quanto maior fosse a quantidade de memória dedicada melhor seriam os resultados de desempenho esperados. É uma conclusão precipitada, pois o *PostgreSQL* utiliza esta memória para efetuar operações e não para *cache* de disco, por exemplo.

Um valor razoável para para o *shared_buffers* é de 25% da memória RAM do sistema (2GB) mas decidiu-se extender os testes até 50% da memória do sistema (4GB). Apesar de valores acima de 40% serem, provavelmente, menos eficazes, pois o *PostgreSQL* conta também com a *cache* do sistema operativo.

Para o *benchamark* foram usados valores de memória RAM entre os 128MB, valor por defeito na configuração, e os 4096MB. Os resultados segue na próxima tabela.

# tamanho (MB)	# pedidos	pedidos/s	lat. média (s)	lat. perct. 99 (s)
128 (Default)	62118	310.5880666032618	1.0220994499090980	1.130365
256	60989	304.9445727406344	1.0224922605223894	1.133323
512	63196	315.9795274241785	1.0210114359611369	1.125195
1024	66182	330.9085519342495	1.0217862895198089	1.110705
2048	70374	351.8684527622802	1.0194107100775853	1.098318
3072	64198	320.9880720942809	1.0198997528427676	1.129971
4096	60497	302.4838197277969	1.0220000570937402	1.134960

Tabela 9: Resultados obtidos para X (X) armazéns

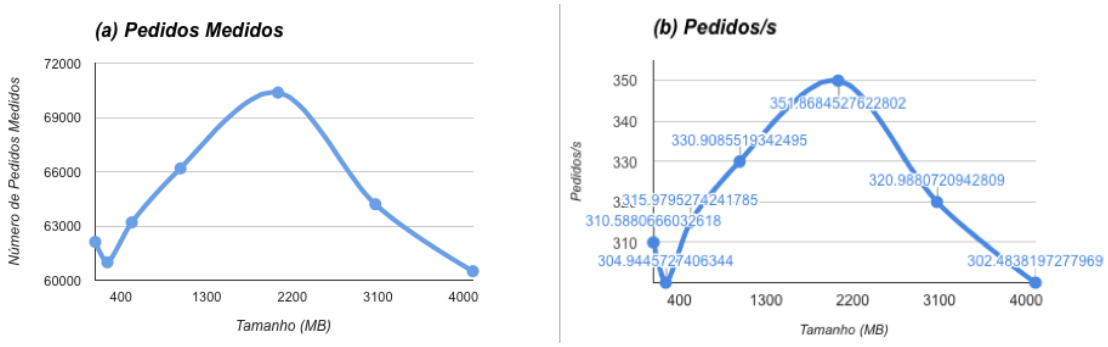

Figura 1: Gráfico correspondente aos valores da tabela anterior

Ao analisar-mos os resultados obtidos, verifica-se que o percentil da latência média e a 99% estão constantemente a diminuir à medida que a quantidade de memória disponível aumenta. Isto acontece devido à ausência de escrita dos resultados para o disco, o que provoca a redução de quantidade de *I/O* para a obtenção de dados. Conclui-se ainda que o melhor resultado aconteceu quando a memória disponível era de **2048MB**, tendo ocorrido mais pedidos por segundo e as latências (média e 99%) apresentarem os resultados mais inferiores. A segunda melhor configuração para esta métrica aconteceu quando a memória disponível era de **1024MB**.

Assim sendo, para esta configuração o valor escolhido será o de **2048MB** por apresentar os melhores resultados tanto a nível de pedidos como de latência.

4.2 Configuração *effective_cache_size*

O *effective_cache_size* é definido como uma estimativa da quantidade de memória disponível para a *cache* do disco. Este parâmetro é utilizado *PostgreSQL Query Planner* para perceber que planos pode utilizar tendo em conta o espaço disponível em memória.

A documentação do *PostgreSQL* afirma que definir 50% da memória total do dispositivo será uma configuração conservadora e que 75% uma configuração agressiva, mas aceitável.

Então para a configuração deste *benchmark* utilizou-se os seguintes parâmetros, 128MB, 512MB, 1024MB, 2048MB, 4096MB e 6144MB.

# tamanho (MB)	# pedidos	pedidos/s	lat. média (s)	lat. perct. 99 (s)
128 (Default)	62118	310.58806660326184	1.022099449909098	1.130365
512	67022	335.1094363325237	1.0208270027602877	1.112753
1024	66987	334.93426967582496	1.0204846512009793	1.109516
2048	66478	332.38835629969145	1.020474478669635	1.112206
4096	59880	299.3983502432346	1.0226386898630595	1.131897
6144	63982	319.9084020175434	1.0210112850958082	1.130562

Tabela 10: Resultados obtidos para X (X) armazéns

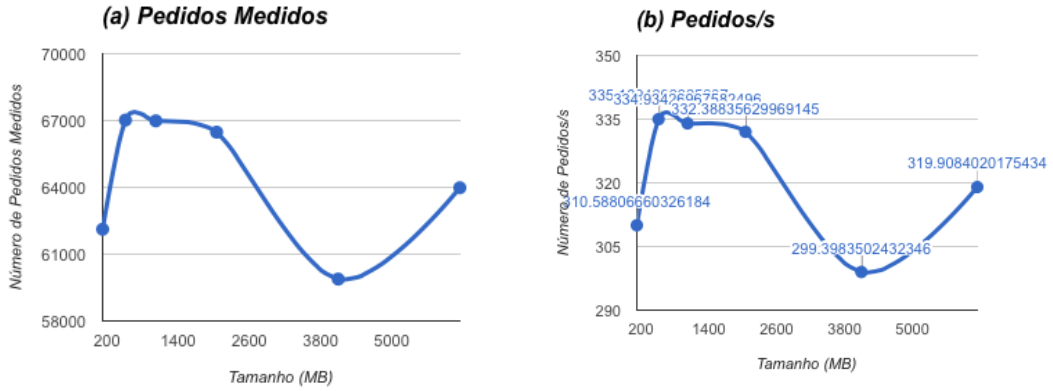

Figura 2: Gráfico correspondente aos valores da tabela anterior

Ao analisar-se os valores obtidos identificasse que na configuração mais conservadora (**4096MB**) e agressiva (**6144MB**) não se obteve os melhores resultados. O número de pedidos medidos foi muito inferior a outros valores assim como a latência a 99%, a latência média praticamente se encontra entre a média dos resultados.

Podemos, então, definir o intervalo de [**512-2048**]**MB** como o que se obteve melhor resultados para esta configuração. O valor **1024MB** obteve melhor resultados no percentil da lantência a 99%. Já o valor **512MB** obteve mais pedidos e pedidos por segundo e um ligeiro aumento nas latências do teste, que influência negativamente o resultado. Com o parâmetro a **2048MB** mantém-se na média dos resultados anteriores, possuíndo o resultado com menor latência média.

Dentro do intervalo definido anteriormente, se fosse necessário escolher um resultado como melhor valor, escolhíamos a configuração com **1024MB** de *effective_cache_size* onde se obteve a menor latência a 99% e valor similares na latência média para o praticamente o mesmo número de pedidos.

4.3 Configuração *checkpoint_segments*

O *PostgreSQL* escreve novas transações na base de dados em ficheiros chamados de segmentos *Write-Ahead Logging (WAL)* que tem um tamanho de 16MB. Nas configurações do *postgresql.conf* o valor atualmente por defeito é de 3 *checkpoint_segments*.

Aumentar o número de *checkpoint_segments* melhora o desempenho do benchmark pois faz com que os *checkpoints* ocorram com menor frequência, obrigando assim a base de dados a uma recuperação mais lenta após uma falha. No entanto o *I/O* ao disco diminui.

Para este *benchmarking* foram utilizados os seguintes valores de *checkpoint_segments*, 3, 9 e 16.

# segmentos	# pedidos	pedidos/s	lat. média (s)	lat. perct. 99 (s)
3 (Default)	62118	310.58806660326184	1.022099449909098	1.130365
9	66254	331.26961583822094	1.0211749849065717	1.113339
16	54564	272.81969749069896	1.0238594005571438	1.103626

Tabela 11: Resultados obtidos para o *checkpoint_segments*.

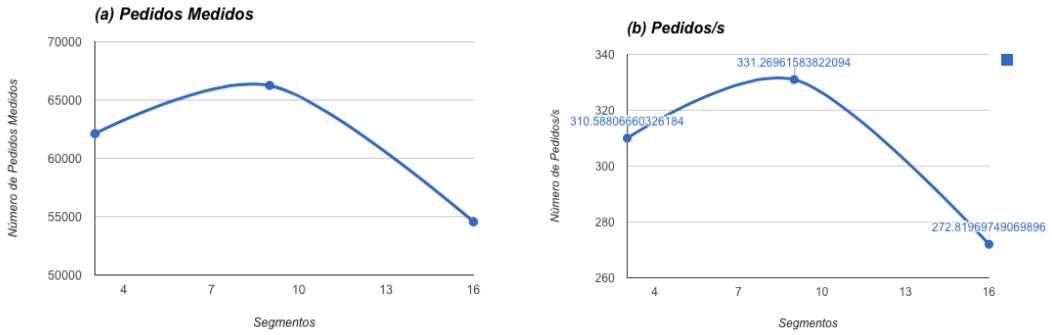

Figura 3: Gráfico correspondente aos valores da tabela anterior

Como foi de esperar, ao aumentar o valor de *checkpoint_segments* o valores de transações aumentaram, mas somente quando o valor foi de 9 segmentos. Quando o teste foi efetuado com **16** segmentos o número de transações dimininuiram abruptamente mas na latência a 99% obteve-se o melhor resultado, ainda que na latência média manteve-se perto dos resultados anteriores. Estes resultados podem-se dever à recuperação mais lenta da base de dados.

Nesta situação a configuração mais indicada seria de **9** segmentos. O *benchmark* obteve o melhore resultado ao nível das transações e de latência média inferior nestas transações.

Normalmente para valores elevados de *checkpoint_segments* é recomendado aumentar o *checkpoint_timeout* por causa dos intervalos de controlo.

4.4 Configuração *autovacuum_naptime*

# tempo (min)	# pedidos	pedidos/s	lat. média (s)	lat. perct. 99 (s)
off	59802	299.0081276096099	1.021858928530818	1.136963
1 [Default]	62118	310.5880666032618	1.022099449909098	1.130365
2	53296	266.4799488385146	1.025329163370609	1.138289
3	61163	305.8148124055486	1.022736959518009	1.124206

Tabela 12: Resultados obtidos para X (X) armazéns

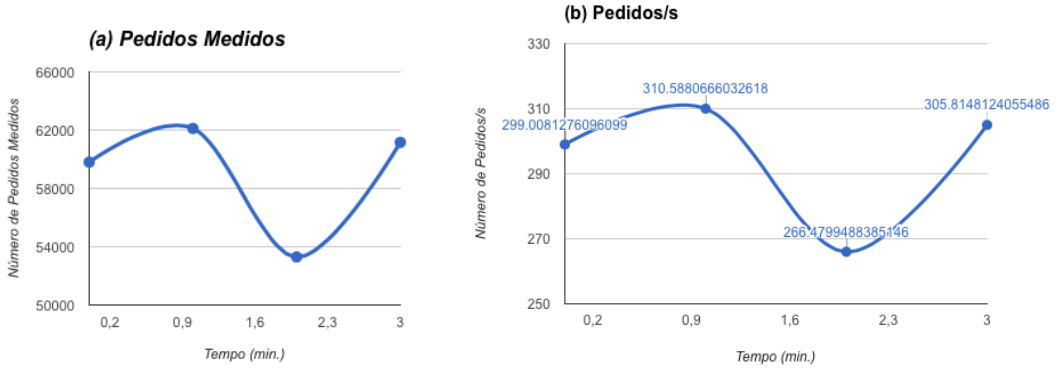

Figura 4: Gráfico correspondente aos valores da tabela anterior

4.5 Configuração *work_mem*

# tamanho (MB)	# pedidos	pedidos/s	lat. média (s)	lat. perct. 99 (s)
1 [Default]	62118	310.5880666032618	1.022099449909098	1.130365
8	62391	311.9513589177765	1.021819045423218	1.130889
32	67109	335.5443187426919	1.019860019833405	1.117067
128	67728	338.6382522575026	1.018854204671627	1.118497
512	67698	338.4895027961542	1.020378764261869	1.118507

Tabela 13: Resultados obtidos para X (X) armazéns

4.6 Configuração do *random_page_cost*

Como o dispositivo utilizado para correr os testes possui um *Solid State Drive (SSD)* decidou-se alterar na configuração do *PostgreSQL* o custo dos acessos aleatórios. Neste teste vamos mostrar os resultados obtidos para a configuração inicial que tem definido um custo igual a 4, e alterações de custo entre 1 e 2.

# custo	# pedidos	pedidos/s	lat. média (s)	lat. perct. 99 (s)
4.0 [Default]	62118	310.58806660326184	1.022099449909098	1.130365
2.0	65295	326.47479814226466	1.0193406799754958	1.116984
1.0	58540	292.69854964502616	1.0225425792791254	1.133753

Tabela 14: Resultados obtidos para 3 tipos diferentes de custos