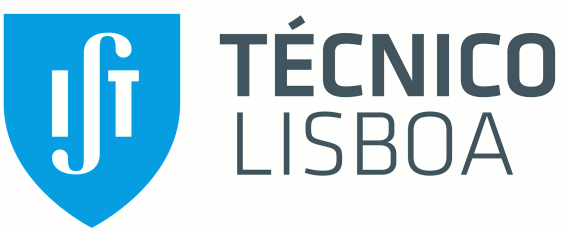

2019/2020

Computação em Nuvem e Virtualização

Checkpoint

Grupo 27

Autores:

André Godinho (Nº 84006)
Sebastião Almeida (Nº 97115)

24/04/2020

1 Análise de métricas

Antes de serem definidas as estruturas de dados do web server, é necessário definir quais as métricas de inetrumentação dos pedidos. Por outras palavras, esta decisão terá em conta não só a informação dada por esta em termos de complexidade de execução do pedido, mas também o acréscimo de overhead criado pelo uso da tool de instrumentação.

Além disso, a análise vai se basear no tempo de execução, pois é o factor mais limitativo. Observou-se que os solvers não apresentam problemas no que toca a alocação de memória.

1.1 Dados adquiridos

Para fazer uma análise detalhada de como as métricas variam em função dos diferentes parâmetros do pedido, usou-se a BIT tool **StatisticsTool** (ativando as flags **-dynamic**, **-load store** e **-alloc**) e realizaram-se diversos testes. Escolheram-se três puzzles 9x9 (01, 03 e 05), três de 16x16 (02, 03 e 04) e um de 25x25 (01). Para cada puzzle, testou-se os três solvers (BFS, CP e DLX) e três valores diferentes de missing values para cada solver. Em concreto, um valor corresponde a existirem entradas a zero apenas no primeiro terço do puzzle, o segundo aos primeiros dois terços do puzzle e o terceiro a existirem elementos a zero em todo o puzzle.

1.2 Análise exploratória

Em primeiro lugar, observou-se de imediato alguma inutilidade de algumas métricas nomeadamente **newarray**, **anew array**, **multianewaray** porque não variavam consoante o input.

Tendo em conta os testes feitos, concluiu-se que as métricas **methods**, **field load**, **field store** e **new** apresentam um overhead mais baixo comparativemente às métricas **instructions**, **blocks**, **regular load** e **regular store**.

Assim sendo, em segundo lugar obteve-se a matriz de correlação das variáveis contínuas (input e métricas). É importante referir que houve uma **correlação de 0.99 entre o tempo de execução e a métrica instructions**, o que faz sentido intuitivamente. Desta forma, durante a análise ir-se-á usar esta métrica para conclusões relativas ao tempo de execução. Para além disso, esta métrica foi a que teve maior correlação com todas as outras e com os parâmetros de input, de modo que é a que consegue representar melhor cada pedido. Contudo, deve-se ter em mente que esta métrica tem um **grande overhead, por isso não é ideal usá-la para a instrumentação**.

A métrica **instructions**, teve uma correlação de **0.82** com a **Area** e **0.84** com os **missing elements**, por isso, **quanto maior a área do puzzle** ou **maior os números em falta**, **maior será o tempo de computação**, como esperado. Das **métricas com menor custo**, a que teve **maior correlação** com a métrica **instructions** foi a **methods (0.86)**. Com efeito, ir-se-á analisar mais detalhadamente a possibilidade de usar-se esta métrica, pois tem um baixo overhead.

1.3 Hipótese: instrumentar com methods

Com o objetivo de se utilizar os **methods** para a instrumentação, foram analisadas em detalhe as tabelas 2, 3 e 4 do anexo.

1.3.1 Análise da tabela 2

A tabela 2 contém os os valores médios de aumento de instruções e métodos quando se varia o número de missing elements entre puzzles. Por exemplo, o número de **instructions** aumentou 2.050 para o solver BFS quando se aumentou de 9 para 16 missing elements e aumentou 1.393 quando se aumentou de 16 para 22 missing elements.

É possível observar que os **methods não acompanham o aumento das instructions** consoante o número de missing elements em todas as áreas. É importante referir que cada valor de missing element para cada área, corresponde aos terços referidos anteriormente. Para além disso, é possível observar que para a **estratégia DLX**, as **insctructions e os methods não variam consosante as entradas a 0 no puzzle**.

Para se ter uma ideia a **average instructions/methods**

1.3.2 Análise da tabela 3

A tabela 3 contém os valores médios de aumento de instruções e métodos quando se varia a área entre puzzles. Por exemplo, o número de **instructions** aumentou em 4.02 ao passar de um puzzle de 16x16 para 25x25 mantendo uma percentagem de missing elements semelhante para a estratégia BFS.

Neste caso é possível observar que diversas **variações foram semelhantes entre os métodos e as instruções**, especialmente para os puzzles onde existiam mais entradas com zeros. Para além disso, o **o aumento é semelhante entre todos os solvers**.

1.3.3 Análise da tabela 4

A tabela 4 contém os valores médios de rácios de instruções e métodos BFS/DLX e CP/DLX. Podemos observar que o **DLX executa mais instruções e métodos** (do que o BFS e CP) à medida que os **puzzles diminuem o número de entradas vazias**. É importante referir que os métodos seguem esta diminuição das instruções.

Em concreto, para os **puzzles de dimensões maiores** e para os **puzzles com mais entradas vazias** (especialmente para os puzzles com missing elements nos 3 terços) as **instructions tendem a aumentar duas vezes relativamente aos methods**. Como vimos anteriormente, **quanto maior a área e quanto maior o número de missing elements maior é o tempo de execução**, pelo que estes **são os casos que requerem maior computação**.

1.4 Conclusão: instrumentar com methods

Tendo em conta o que foi dito na secção 1.3.1 e 1.3.3., conclui-se que **usar os methods não chega para representar suficientemente bem o custo de cada pedido**.

Uma vez que as **instruções aumentam pelo menos duas vezes mais rapidamente que os métodos nos pedidos mais exigentes computactionalmente para o solver BFS e CP**, ao usar-se **apenas os methods** como custo dos pedidos irá haver um **mapeamento errado entre os pedidos**. Por outras palavras, **pedidos com duas ou mais intruções irão ter os mesmos métodos**.

O exemplo na tabela 1 representa o que está a ser explicado.

Puzzle	% missing	Solver	Instructions	Methods
9x9_05	19.75%	DLX	526652934	3547
16x16_02	15.23%	CP	1727241746	3305

Tabela 1: Exemplo representativo de dois testes onde um é mais exigente computactionalmente do que o outro (três vezes mais instructions), mas têm um número de methods semelhante

2 Solução: função de custo

De facto, os **methods** não parecem ser um candidato fiável para se medir o custo dos pedidos. No entanto, observou-se durante toda a análise que ainda assim existe informação útil a ser detetada por esta métrica. Aliás, a **alta correlação com as instructions**, as **variações do número de métodos** em função da **variação da área dos puzzles** referida na secção 1.3.2. e o **seguimento** (diminuição) **existente dos métodos face às instructions** na secção 1.3.3 indicam isso mesmo. Para além disso, esta métrica acrescenta um overhead extremamente baixo quando comparada às outras métricas, por isso faz sentido criar uma função de custo que usará a informação que esta métrica fornece.

Assim sendo, propõe-se uma função de custo que visa contornar os pontos negativos descritos na secção 1.3.1. e 1.3.3, usando os parâmetros de cada pedido.

2.1 Solução para 1.3.1.

Em primeiro lugar, para penalizar os pedidos com maior percentagem de entradas vazias **para o CP e BFS**, propõe-se que esta função de custo (que deixa de ser medida em methods) aplique a seguinte expressão:

$Cost = Methods + Methods \cdot 2 \cdot \%missingelements$

Multiplicar por dois penalizará ainda mais puzzles com um maior número entradas vazias. Este número pode ser ajustado, contudo para este *checkpoints* propomos este valor. Na tabela 2 é possível observar uma **melhoria face a usar os methods**.

2.2 Solução para 1.3.3.

Em segundo lugar, ir-se-á **aumentar o valor dos methods** de modo que se aproxime do **crescimento de instruções para os solvers BFS e CP**.

$Cost_{BFS} = 1.8 \cdot (Methods + Methods \cdot 2 \cdot \%missingelements)$
$Cost_{DLX} = 1.8 \cdot (Methods + Methods \cdot 2 \cdot \%missingelements)$
$Cost_{DLX} = Methods$

Na tabela 4 é possível observar uma **melhoria face a usar os methods**, pois os rácios estão mais próximos aos das instructions.

2.3 Conclusão: função de custo

Após esta alteração, calculou-se o custo para todos os testes realizados. A **correlação entre instructions e o custo é de 0.98** (para os methods era de 0.86), o que ajuda a comprovar as soluções apresentadas.

Desta forma, ir-se-á implementar a aplicação desta função de custo para a obtenção dos custos dos pedidos.

3 Estruturas de dados

Para este *checkpoint* as **instâncias estão a escrever os parâmetros dos pedidos e as métricas localmente** para um *log* através dum *concurrent hashmap*. Optou-se por um concurrent hashmap para **facilitar a sincronização entre threads e para a modificação dos valores ser *thread-safe***. Por outras palavras, o **WebServer** contém um objeto da classe **MeasurementsManager**, cujo atributo é esta estrutura. As **chaves** do *concurrent hashmap* são o **thread ID** de modo que seja possível a escrita na base de dados paralelamente. Os ***items*** associados às chaves são uma **lista que conterá objetos Measurement** que **encapsulam as métricas e os parâmetros dos pedidos** que estão guardados num objeto da classe **Request**. Assim, caso sejam feitos diversos pedidos simultaneamente, cada thread irá executar o solver instrumentalizado e a escrita nesta estrutura é feita acedendo a uma chave diferente. É importante referir que ainda **não está implementada a função de custo**, mas assim que estiver ir-se-á alterar a classe Measurement, guardando para cada pedido o custo do mesmo.

Quando for implementada a DynamoDB será estudada a hipótese de usar caches locais para cada instância, pelo que poder-se-á aproveitar parte do que já está feito, nomeadamente a utilização de listas para guardar as informações dos pedidos, faltando apenas serializar as mesmas.

É importante referir que o método que implementa a escrita para os *logs* é *synchronized*, assim como a inserção no *concurrent hashmap* para **garantir que estas operações sejam *thread-safe*, evitando *race conditions***. Para além disso, os métodos invocados pela BIT tool implementada são estáticos de modo que não sejam criadas variáveis durante a instrumentalização e que toda a implementação se baseou em utilizar *getters* e *setters*. Em concreto, estes são usados para manipular os dados entre os objetos criados para os pedidos no *handler* do WebServer e os solvers instrumentados pela BIT tool.

4 Load balancer

4.1 Implementação actual

Para ser possível a distribuição dos pedidos pedidos por diversas instâncias, **foi criado um novo *endpoint* para receber health checks pela AWS**. Este *endpoint* é dado por \health e retorna o statuscode 200 com a mensagem "OK". Assim é possível verificar se as instâncias estão *healthy*.

De seguida, criou-se um load balancer com as configurações definidas no *README*. Depois, enviou-se pedidos para o DNS do load balancer e port 80, tendo estas mensagem sido recebidas pelas diversas instâncias (criadas pelo autoscaler) e respondido as respostas pretendidas.

4.2 Próxima etapa

Para se decidir a abordagem a aplicar no que toda ao *load balancing*, deve-se ter em consideração que as instâncias a que se tem acesso na AWS são instâncias iguais. Isto é, não existem instâncias com maior poder computacional do que outras. Assim, não é necessário aplicar *weights* para que certos servidores recebam mais pedidos do que outros devido a diferentes capacidades computactionais.

Para haver uma distribuição de pedidos, escolheu-se o algoritmo *least load*. Por outras palavras, uma vez recebido um pedido, o load balancer irá **calcular o custo do mesmo através de pedidos semelhantes guardados no *Metrics Storage System***, ou seja, ir-se-á aceder ao número de *Methods* medidos nesse pedido semelhante e calcular o custo do novo pedido. De seguida, **escolhe-se o servidor que tenha menor carga**, que é dado para cada servidor pela **soma dos custos dos pedidos nesse servidor**. Ao ser atribuído o pedido a esse servidor, atualiza-se o novo valor de carga do mesmo. Quando o pedido for resolvido e instrumentado, **retifica-se o valor da carga retirando o custo da previsão**. À medida que serão feitos mais pedidos, o *Metrics Storage System* será capaz de fornecer previsões de custos mais acertadas, o que irá ter um impacto positivo no *load balancing*.

Quando o *Metrics Storage System* não tem pedidos semelhantes, irá ser feita uma previsão (aproximada) do custo, esta abordagem será analisada na próxima etapa do projecto.

É importante acrescentar que o load balancer irá realizar *health checks* às instâncias que estão ligadas através do novo *endpoint* criado para saber se estão em condições de receber pedidos.

5 Auto-scaling

Para este *checkpoint*, que funciona com base em instâncias da AWS, vamos utilizar uma estratégia de *horizontal scaling*. Ou seja, é através da abertura e fecho de instâncias que vamos escalar a capacidade de processamento do nosso sistema à demanda que os pedidos que recebemos representam.

5.1 Implementação Actual

Neste ponto intermédio do projecto, o nosso sistema utiliza a solução de auto-scaling fornecida pela própria AWS. O número de instâncias activas varia entre 1 e 5, sendo que é dado o "alarme"para a abertura de uma nova instância quando a média de utilização de CPU entre as instâncias é superior a 40% durante mais de 60 segundos. No sentido oposto, é fechada uma instância quando essa média de utilização de CPU é inferior a 20% durante mais de 60 segundos. As configurações realizadas foram decidads para verificar o funcionamento correto entre o load balancer, autoscaler e as instâncias criadas. Para além disso, também se teve em conta dar um tempo suficiente ao *grace period* de modo que as instâncias fossem consideradas como *healthy*. Estes parâmetros de configuração irão ser analizados em maior detalhe para a entrega final.

5.2 Próxima etapa

O plano para a versão final do nosso sistema é que o processo dinâmico de abertura e fecho de novas instâncias seja controlado directamente pelo nosso código, deixando a solução *standard* da AWS.

6 Anexo

Área	Missing Elements	Estratégia	Aumento Instructions	Aumento Methods	Aumento Custo	I	M	C
81	9	BFS	-	-	-	1.098e8	643	1415
81	9	CP	-	-	-	9.745e7	625	1376
81	9	DLX	-	-	-	5.267e8	3548	3548
81	16	BFS	2.050	1.317	1.505	2.251e8	848	2131
81	16	CP	1.993	1.154	1.318	1.942e8	722	1814
81	16	DLX	1.027	1.007	1.007	5.407e8	3572	3572
81	22	BFS	1.393	1.179	1.304	3.136e8	1000	2778
81	22	CP	1.488	1.134	1.254	2.889e8	818	2275
81	22	DLX	1.022	1.006	1.006	5.528e8	3593	3593
256	23	BFS	-	-	-	3.944e8	1851	3932
256	23	CP	-	-	-	3.884e8	1810	3846
256	23	DLX	-	-	-	1.766e9	12755	12755
256	39	BFS	1.933	1.333	1.474	7.721e8	2469	5795
256	39	CP	2.666	1.384	1.529	1.035e9	2505	5880
256	39	DLX	1.016	1.007	1.007	1.795e9	12839	12839
256	54	BFS	2.436	1.723	1.879	1.881e9	4254	10890
256	54	CP	1.562	1.232	1.343	1.617e9	3085	7897
256	54	DLX	1.015	1.007	1.007	1.822e9	12932	12932
625	46	BFS	-	-	-	1.606e9	5289	10929
625	46	CP	-	-	-	1.297e9	4585	9474
625	46	DLX	-	-	-	5.237e9	36532	36532
625	80	BFS	1.597	1.292	1.413	2.565e9	6832	15445
625	80	CP	1.898	1.252	1.370	2.462e9	5741	12979
625	80	DLX	1.007	1.003	1.003	5.276e9	36642	36642
625	114	BFS	1.508	1.304	1.416	3.868e9	8910	21875
625	114	CP	1.655	1.282	1.392	4.074e9	7361	18072
625	114	DLX	1.007	1.004	1.004	5.316e9	36784	36784

Tabela 2: Valores médios de aumento de instruções, métodos e do custo estimado para todos os testes feitos segundo a área, missing elements e estratégia

Terços c/0	Área	Estratégia	Aumento instructions	Aumento methods	Aumento Custo	Instructions	Methods	Cost
1	625	BFS	4.02	2.86	2.78	1.606e9	5289	10929
1	625	CP	3.34	2.53	2.46	1.297e9	4585	9474
1	625	DLX	2.97	2.86	2.86	5.237e9	36532	36532
1	256	BFS	3.64	2.88	2.78	3.992e8	1852	3933
1	256	CP	3.99	2.89	2.80	3.884e8	1811	3846
1	256	DLX	3.35	3.59	3.59	1.766e9	12755	12755
1	81	BFS	-	-	-	1.098e8	644	1416
1	81	CP	-	-	-	9.745e7	626	1376
1	81	DLX	-	-	-	5.266e8	3548	3548
2	625	BFS	3.32	2.77	2.67	2.565e9	6832	15446
2	625	CP	2.38	2.29	2.21	2.462e9	5741	12979
2	625	DLX	2.94	2.85	2.85	5.276e9	36642	36642
2	256	BFS	3.43	2.91	2.72	7.721e8	2469	5795
2	256	CP	5.33	3.47	3.24	1.036e9	2505	5881
2	256	DLX	3.32	3.59	3.59	1.795e9	12839	12839
2	81	BFS	-	-	-	2.251e8	848	2131
2	81	CP	-	-	-	1.942e8	722	1814
2	81	DLX	-	-	-	5.407e8	3573	3573
3	625	BFS	2.06	2.09	2.01	3.868e9	8910	21876
3	625	CP	2.52	2.39	2.29	4.074e9	7361	18073
3	625	DLX	2.92	2.84	2.84	5.316e9	36784	36784
3	256	BFS	6.00	4.26	3.92	1.881e9	4255	10890
3	256	CP	5.60	3.77	3.47	1.617e9	3085	7897
3	256	DLX	3.30	3.60	3.60	1.823e9	12933	12933
3	81	BFS	-	-	-	3.136e8	1000	2778
3	81	CP	-	-	-	2.889e8	819	2275
3	81	DLX	-	-	-	5.529e8	3593	3593

Tabela 3: Valores médios de aumento de instruções, métodos e do custo estimado para todos os testes feitos segundo a percentagem de missing elements, área, e estratégia

Terços com 0	Área	Estratégia	Rácio Instructions	Rácio Methods	Rácio Custo	Instructions	Methods	Cost
1	625	BFS	3.26	6.91	3.34	1.606e9	5289	10929
1	625	CP	4.04	7.97	3.86	1.297e9	4585	9474
1	625	DLX	1	1	1	5.237e9	36532	36532
1	256	BFS	4.42	6.89	3.24	3.992e8	1852	3933
1	256	CP	4.55	7.04	3.32	3.884e8	1811	3846
1	256	DLX	1	1	1	1.766e9	12755	12755
1	81	BFS	4.80	5.51	2.51	1.098e8	644	1416
1	81	CP	5.4	5.67	2.58	9.745e7	626	1376
1	81	DLX	1	1	1	5.266e8	3548	3548
2	625	BFS	2.06	5.36	2.37	2.565e9	6832	15446
2	625	CP	2.14	6.38	2.82	2.462e9	5741	12979
2	625	DLX	1	1	1	5.276e9	36642	36642
2	256	BFS	2.33	5.20	2.22	7.721e8	2469	5795
2	256	CP	1.73	5.13	2.18	1.036e9	2505	5881
2	256	DLX	1	1	1	1.795e9	12839	12839
2	81	BFS	2.40	4.21	1.68	2.251e8	848	2131
2	81	CP	2.78	4.95	1.97	1.942e8	722	1814
2	81	DLX	1	1	1	5.407e8	3573	3573
3	625	BFS	1.37	4.13	1.68	3.868e9	8910	21876
3	625	CP	1.3	5.00	2.04	4.074e9	7361	18073
3	625	DLX	1	1	1	5.316e9	36784	36784
3	256	BFS	0.97	3.04	1.19	1.881e9	4255	10890
3	256	CP	1.13	4.19	1.64	1.617e9	3085	7897
3	256	DLX	1	1	1	1.823e9	12933	12933
3	81	BFS	1.76	3.59	1.29	3.136e8	1000	2778
3	81	CP	1.91	4.39	1.58	2.889e8	819	2275
3	81	DLX	1	-	1	5.529e8	3593	3593

Tabela 4: Rácios médios de instruções, métodos e do custo DLX/BFS e DLX/CP estimado para todos os testes feitos segundo a percentagem de missing elements, área, e estratégia