Análise de Desempenho do Ambiente de Execução Interna do Python no Banco de Dados SQL Server

Gustavo Cordeiro Galvão Van Erven

Brasília, Distrito Federal

Abstract

Esse trabalho busca comparar o desempenho entre a execução do algoritmo *K-means* implementado em Python em um ambiente externo e dentro do banco SQL Server. Foram realizadas 11 execuções com 50 repetições cada para subsidiar a comparação. Foi verificado que, apesar dos dados estarem próximos ao processamento na execução interna, o ambiente externo ainda apresentou melhor resultado.

Keywords: Banco de Dados, SQL Server, Python, Desempenho

1. Introdução

Inteligência Artificial e processamento de grandes volumes de dados são áreas que vêm se destacando atualmente, inclusive pela expectativa de revolucionar diversas áreas da sociedade [1, 2].

No centro dessa evolução estão os dados, que trazem uma enorme diversidade de informações de várias áreas diferentes, como clientes de empresas, sensores de clima, usuários de programas de governo, etc. Muitos desses dados são armazenados em bancos de dados dos mais diversos tipos e são provessados por ferramentas que realizam atividades como busca, filtragem, análise estatística, mineração de dados, entre outras [3]. Entre as diversas ferramentas de análise, está o uso da linguagem de programação Python.

Entretanto, para serem processados, esses dados devem ser lidos de uma origem, como um arquivo ou banco de dados, e carregados na ferramenta. Para diminuir a distância entre os dados e o processamento, alguns sistemas gerenciadores de banco de dados (SGDB) adicionaram serviços internos em diversas linguagem, inclusive Python. Dessa forma, o próprio SGDB pode

iniciar uma seção passando para ela os dados sem necessidade de utilizar outros mecanismos como *Open Database Connectivity* (ODBC), por exemplo.

Dessa forma, espera-se que o desempenho da execução do mesmo código dentro do banco de dados seja equivalente ou superior ao executado utilizando o procedimento convencional de buscar do banco para outro servidor e executar a análise. Esse trabalho busca quantificar essa diferença executando o algoritmo *K-means* [4] do Python. Apesar da expectativa de melhora, a execução externa permaneceu superior comparada ao componente interno do SGDB como será apresentado. Ainda assim, esse mecanismo permite simplificar diversos tipos de operações, por manter o processamento ao lado dos dados quando utilizado um banco de dados relacional.

O restante desse artigo é dividido da seguinte forma: A Seção 2 apresenta alguns trabalhos relacionados. A Seção 3 detalha a metodologia do trabalho. A Seção **??** descreve a análise sobre os dados coletados. Por fim, a Seção 5 apresenta as conclusões finais desse artigo.

2. Trabalhos relacionados

Trazer o dado de forma mais eficiente para processamento já é uma preocupação e tema de estudo e se popularizou bastante a partir de softwares de *Big Data* como Hadoop [5]. Esse sistema utiliza um sistema de arquivos distribuído chamado *Hadoop Distributed File System* [6] (HDFS) para paralelizar a entrega dos dados para diversos processos aumentando a vazão entre disco e memória do *cluster*.

Alguns outros bancos trouxeram algoritmos específicos para seu código interno, como o BayesDB [7], que permite completar dados faltantes utilizando estatística baysiana para inferir com certo intervalo de confiabilidade.

Outro banco que permite executar algoritmos de aprendizagem de máquina é o *Machine Learning Database* [1] (MLDB) que permite tratar os dados e salvar o modelo criado no banco, inclusive de imagens.

Esses exemplos demonstram uma tendencia de aproximar o processamento da fonte dos dados aproveitando a infraestrutura de armazenamento já existente que é o SGDB.

[1] https://mldb.ai/

3. Metodologia

Para esse trabalho foram definidos que dois servidores seriam utilizados. Ambas as máquinas com mesma configuração de *hardware*, porém uma com Windows, onde o SQL Server foi instalado, e outra com Linux para simular uma aplicação que acesse dados no banco. Dois fatores foram definidos também: o tamanho da amostra e se a execução é interna (no banco) ou externa (em outra aplicação).

Para esse experimento, foi realizada a execução de uma *clusterização* com parâmetro de k$\bar{2}$0 grupos. Dois *scripts* foram montados, um para o SQL Server e outro simplesmente em Python, que capturaram o tempo de início e fim de cada rodada e o tempo de execução apenas do comando principal a ser executado, ou seja, a parcela do tempo total gasto para executar o agrupamento.

Para permitir uma análise mais precisa, cada execução com um conjunto de amostra foi repetida 50 vezes e foram variadas as cargas de 1000 registros, dobrando-se a cada execução, até atingir 1.024.000 de registros. Dessa forma, foram levantadas 11 cargas diferentes para cada um dos dois fatores e geradas as estatísticas de média, desvio padrão e intervalos de confiança. Também foi realizada uma análise fatorial para verificar o impacto de cada um dos fatores, além de uma regressão sobre a evolução das médias de cada fator.

4. Análise

A análise os dados armazenados em CSV foram carregados em uma notebook do Python e foram calculadas as médias de cada execução para cada um dos dois fatores (tamanho da amostra e local de execução)[2]. A Tabela 1 sumariza os resultados obtidos. Primeiramente é apresentado o tamanho da amostra, seguido das médias para a execução dentro do SQL Server e depois para a execução no ambiente externo. Por fim, a proporção equivalente à execução do Python com relação ao tempo total também é calculada. Todos os tempos estão em milissegundos e o intervalo de confiança de todas as médias não incluem o zero.

Diferentemente do esperado, os tempos de execução são superiores no ambiente externo quando comparado com a execução dentro do banco de dados. Isso pode ter ocorrido por conta de uma sobrecarga de operações

[2]Disponível em: https://github.com/gvanerven/sqlserver_python

Tam. Amostra	**Média MSSQL Python**	**SEM MSSQL Python**	**Média MSSQL Total**	**SEM MSSQL Total**	**Média Ext. Python**	**SEM Ext. Python**	**Média Ext. Total**	**SEM Ext. Total**	**MSSQL** $100 \times \frac{Python}{Total}$	**Ext.** $100 \times \frac{Python}{Total}$
1000	0.123439	0.002058	2175.72	66.146825	0.004804	0.000059	**0.025265**	0.000330	0.005673	19.013149
2000	0.174689	0.010448	2159.34	11.438339	0.005841	0.000074	**0.045689**	0.000965	0.008090	12.783224
4000	1.002512	0.121612	2997.86	121.257372	0.006134	0.000111	**0.082430**	0.000952	0.033441	7.440831
8000	1.457830	0.135186	3475.92	135.396255	0.007714	0.000092	**0.171707**	0.002071	0.041941	4.492800
16000	1.829398	0.113231	3881.92	113.141684	0.015518	0.000538	**0.301726**	0.002303	0.047126	5.143140
32000	3.013475	0.100778	5137.88	100.511393	0.016306	0.000115	**0.572136**	0.001617	0.058652	2.850064
64000	4.971625	0.076094	7206.00	75.427689	0.022154	0.000156	**1.167824**	0.005852	0.068993	1.897030
128000	9.563245	0.087227	11982.98	88.434435	0.040379	0.000604	**2.358322**	0.011923	0.079807	1.712181
256000	19.192428	0.084866	22029.92	85.575845	0.061625	0.000942	**4.620593**	0.024592	0.087120	1.333700
512000	42.789599	0.109536	46473.32	109.819159	0.170140	0.001823	**9.521799**	0.073421	0.092073	1.786849
1024000	105.165695	0.144261	110708.30	143.106215	0.293033	0.005766	**18.544739**	0.100246	0.094994	1.580143

Tabela 1: Dados das médias dos tempos em milissegundos de execução para os dois fatores.

que talvez o banco precise realizar para encadear os dados com a chamada interna do comando. O teste **t-Student** entre os tempos de execução total externo e interno resultou em um $p - valor = 2.4573 \times 10^{-45}$ e o t-Student entre os tempos do *K-means* do Python foi de $p - valor = 3.3043 \times 10^{-37}$, ou seja, ambos bem abaixo do $p - valor = 0.05$ para 90% de intervalo de confiança. O tempo de execução do comando Python dentro do sistema também surpreendeu pela diferença com relação ao ambiente externo. A Tabela 2 apresenta a razão entre as médias para cada tamanho de amostra.

Tamanho da Amostra	**Razão das Médias Totais**	**Razão das Médias do Python**
1000	86114.695321	25.696356
2000	47261.638887	29.909805
4000	36368.338310	163.448297
8000	20243.271353	188.973453
16000	12865.706386	117.887246
32000	8980.172447	184.804978
64000	6170.449176	224.412268
128000	5081.146381	236.838623
256000	4767.769222	311.439772
512000	4880.728799	251.496192
1024000	5969.795471	358.886295

Tabela 2: Razão entre as médias internas e externas.

Percebe-se que as médias internas superam em ordem de grandeza até 10^4 as médias externas. O p-valor do teste X^2 para as médias apenas do

comando do Python foi de $p - valor = 5.0985243504669014^{-35}$, ou seja, um valor extremamente abaixo de $p - valor = 0.05$ confirmando a diferença entre as execuções internas e externas.

4.1. Projeto 2^{kr} fatorial com k = 2 e r = 50

Para verificar o impacto dos fatores no experimento, foi realizado um projeto 2^{kr} fatorial com k = 2 e r = 50. Considerou-se o fator A como o tamanho da amostra e o fator B como local de execução. A amostra de 1000 registro foi definida como -1 e a superior de 1. Como uma amostra pode ser bem maior de um grupo de execução para o outro, foi feita a análise utilizando a amostra superior primeiramente como 16.000 e depois a final de 1.024.000 registros. O fator B ficou com nível -1 quando a execução é externa e 1 quando interna.

Projeto 1:

Qtd 1000 -> -1 (A), Qtd 16000 -> +1

Exec Ext -> -1 (B), Exec Int -> +1

Projeto 2:

Qtd 1000 -> -1 (A), Qtd 1024000 -> +1

Exec Ext -> -1 (B), Exec Int -> +1

O resultado do Projeto 1 é apresentado na Tabela 3. Pode-se observar que o fator considerando onde o comando foi executado explica 78.58% da variação. O tamanho da amostra explica apenas 6.29%, menos que o a variação pelo erro de 8.82%. A estatística F calculada para esse projeto foi de 34.94 para os graus de liberdade 3 e 147, bem acima na tabela do valor de infinito de 3.1161.

Fator	**Proporção explicada**
Tamanho da amostra (A)	0.0629278904135
Local de execução (B)	0.78594401553
Interação dos fatores (AB)	0.0628853486447
Erro experimental	0.0882427454119

Tabela 3: Resultado do Projeto 1 (Amostra máxima de 16.000).

Já para o segundo projeto, como a distância entre a quantidade inicial e final é bem mais relevante, os valores das proporções se alteraram significativamente. A Tabela 4 apresenta os resultados. A estatística F para esse projeto ficou em 49270.10.

Fator	Proporção explicada
Tamanho da amostra (A)	0.325177950701
Local de execução (B)	0.349323866985
Interação dos fatores (AB)	0.325174787005
Erro experimental	0.000323395307937

Tabela 4: Resultado do Projeto 2 (Amostra máxima de 1.024.000).

Observa-se que agora os valores das proporções estão quase igualmente distribuídos, inclusive na interação entre os fatores e o erro caiu para 0.032%, bem abaixo do anterior. Isso se deve, como dito anteriormente, pela distância entre o valor inicial e final dos fatores.

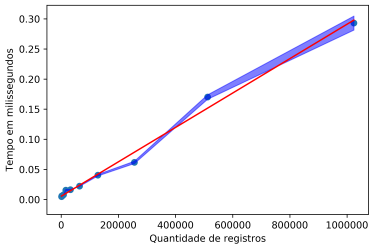
4.2. Análise gráfica e regressão

Pode-se analisar o comportamento do tempo de execução médio a partir do tamanho da amostra. Utilizando os dados das execuções, foram construídos os gráficos da Figura 1. Os resultados obtidos foram plotados como ponto e a espessura do preenchimento acompanhando o ponto indica o intervalo de confiança da média indicada pelo mesmo. A linha reta indica uma regressão simples calculada a partir dos dados plotados no formato $y = b_0 + b_1 \times x$.

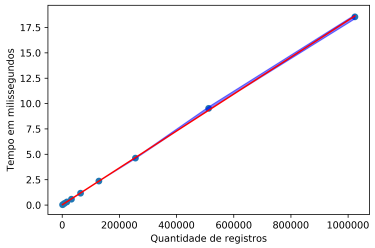
Os gráficos da Figura 1a e Figura 1c apresentam os resultados da execução apenas do comando do Python, no ambiente externo e interno respectivamente. Como o *K-means* realiza o cálculo de distância entre todos os pontos para cada *centroid* até a convergência, espera-se uma complexidade seja $O(kn)$. Nesse experimento, k foi mantido constante, e assim é esperado um crescimento linear no tempo de execução para o algoritmo isolado. Observando o gráfico, percebe-se que existe, para os pontos medidos, uma tendencia de se manter próximo da reta de regressão. Considerando a aparente aderência na execução total, pode-se supor que a taxa de transferência dos dados entre o banco e a aplicação seja linear com relação ao número de registros, entretanto o valor máximo da amostra pode se ainda baixo para evidenciar uma tendencia não linear.

A Tabela 5 apresenta os coeficientes de determinação e intervalos de confiança a 95% para cada uma. Para as execuções, o R^2 ficou superior a 0.97%, entretanto, percebe-se, pelo intervalo de confiança, que o coeficiente b_1 não é significativo para qualquer das regressões.

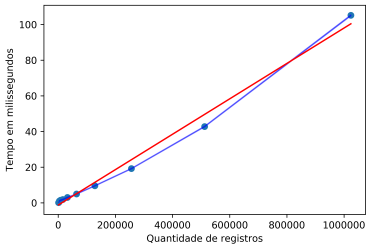
A Figura 1c e a Figura 1d apresentam os dados coletados na execução interna com o tempo do Python e Total respectivamente. Diferente da exe-



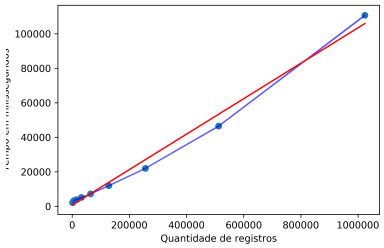
(a) Execução Externa apenas Python.



(b) Execução Externa Total.



(c) Execução Interna apenas Python.



(d) Execução Interna Total.

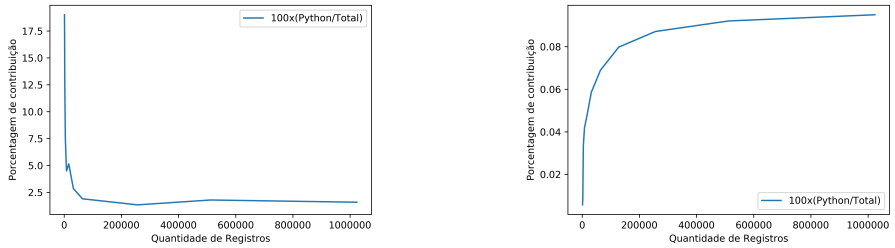
Figura 1: Evolução das médias com relação à quantidade de registros.

cução externa, o tempo do *K-means* e total dentro do banco já apresentam um leve desvio abaixo da reta, sugerindo uma complexidade quadrática no intervalo observado. Provavelmente, os mecanismos internos para controlar e vincular os dados e executar do comando internamente devam explicar essa variação.

Regressão	**Coeficientes**	R^2	**IC de 95%**
Externa Python	b_0=0.0053, b_1=2.8573e-07	0.9917	IC_0=(-0.0017, 0.0123), IC_1=(2.6608e-07, 3.0538e-07)
Externa Total	b_0=0.0189, b_1=1.8175e-05	0.9998	IC_0=(-0.0410, 0.0787), IC_1=(1.8007e-05, 1.8343e-05)
Interna Python	b_0=-1.2650, b_1=9.9267e-05	0.9890	IC_0=(-4.0693, 1.5393), IC_1=(1.8007e-05, 1.8343e-05)
Interna Total	b_0=728.7589, b_1=0.1027	0.9896	IC_0=(-2097.5945, 3555.1123), IC_1=(9.1400e-05, 0.0001)

Tabela 5: R^2 e Intervalos de Confiança de 95% das Regressões.

Por fim, a Figura 2 apresenta a contribuição da execução do *K-means* com relação ao tempo Total. A Figura 2a mostra como o tempo das demais



(a) Externo: Proporção do tempo do Python com relação ao Total.

(b) Interno: Proporção do tempo do Python com relação ao Total.

Figura 2: Proporções para as execuções Externas e Internas entre Python e Total.

operações, incluindo a consulta e recebimento dos dados, torna-se dominante a medida que o número de registros aumenta na execução externa. Esse resultado era esperado, imaginando que o tempo de transmissão poderia se tornar o gargalo.

Por outro lado, a Figura 2b apresenta uma tendência de evolução na contribuição do tempo do *K-means* no ambiente interno até uma aparente estabilização após a marca dos 400.000 registros. Essa sobrecarga dentro do comando pode ser devida à passagem dos dados entre o banco e o comando, e por isso poderia acompanhar a complexidade na obtenção dos dados. Assim como a regressão, esses gráficos precisam ser expandidos para realmente verificar se essas tendências se mantem ou se são regiões isoladas.

5. Conclusão

O presente trabalho apresenta uma análise comparativa entre a execução do algoritmos *K-means* implementado em Python em uma ambiente externo e dentro do banco de dados SQL Server.

Foi verificado que, apesar da execução interna manter os dados mais próximos da aplicação, o ambiente externo, buscando os registros da mesma instância, tiveram um desempenho melhor em uma ordem de grandeza de até 10^4.

Uma possível explicação para esse fenômeno é a sobrecarga interna para gerir e vincular os dados entre os mecanismos internos de dados e de execução dentro do banco de dados.

Embora que a execução internar ainda não tenha um desempenho superior, ao menos para esse caso testado, a possibilidade de execução de ambi-

entes completos de linguagem de programação permite simplificar operações por manter o dado próximo ao processamento e pode ser tornar uma ferramente essencial futuramente.

[1] C. Yang, Q. Huang, Z. Li, K. Liu, F. Hu, Big data and cloud computing: innovation opportunities and challenges, International Journal of Digital Earth 10 (2017) 13–53.

[2] S. Makridakis, The forthcoming artificial intelligence (ai) revolution: Its impact on society and firms, Futures 90 (2017) 46–60.

[3] R. A. Rodrigues, L. A. Lima Filho, G. S. Gonçalves, L. F. Mialaret, A. M. da Cunha, L. A. V. Dias, Integrating nosql, relational database, and the hadoop ecosystem in an interdisciplinary project involving big data and credit card transactions, in: Information Technology-New Generations, Springer, 2018, pp. 443–451.

[4] A. K. Jain, Data clustering: 50 years beyond k-means, Pattern recognition letters 31 (2010) 651–666.

[5] T. White, Hadoop: The definitive guide, "O'Reilly Media, Inc.", 2012.

[6] K. Shvachko, H. Kuang, S. Radia, R. Chansler, The hadoop distributed file system, in: Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on, Ieee, pp. 1–10.

[7] V. Mansinghka, R. Tibbetts, J. Baxter, P. Shafto, B. Eaves, Bayesdb: A probabilistic programming system for querying the probable implications of data, arXiv preprint arXiv:1512.05006 (2015).