

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 processados 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 linguagens, inclusive Python. Dessa forma, o próprio SGDB pode

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

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

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

32 2. Trabalhos relacionados

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

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

42 Outro banco que permite executar algoritmos de aprendizagem de má-
43 quina é o *Machine Learning Database*¹ (MLDB) que permite tratar os dados
44 e salvar o modelo criado no banco, inclusive de imagens.

45 Esses exemplos demonstram uma tendencia de aproximar o processa-
46 mento da fonte dos dados aproveitando a infraestrutura de armazenamento
47 já existente que é o SGDB.

¹<https://mldb.ai/>

48 3. Metodologia

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

55 Para esse experimento, foi realizada a execução de uma *clusterização*
56 com parâmetro de k20 grupos. Dois *scripts* foram montados, um para o SQL
57 Server e outro simplesmente em Python, que capturaram o tempo de início
58 e fim de cada rodada e o tempo de execução apenas do comando principal
59 a ser executado, ou seja, a parcela do tempo total gasto para executar o
60 agrupamento.

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

68 4. Análise

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

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

²Disponível 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 Python Total	Ext. 100 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.

81 que talvez o banco precise realizar para encadear os dados com a chamada
82 interna do comando. Utilizando um teste X^2 (*chi-square*) para comparar
83 as médias de tempo total foi obtido um p-valor de 0.0 (provavelmente um
84 valor muito pequeno que ficou em zero por conta da precisão de onde foi
85 calculado). O tempo de execução do comando Python dentro do sistema
86 também surpreendeu pela diferença com relação ao ambiente externo. A
87 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.

88 Percebe-se que as médias internas superam em ordem de grandeza até
89 10^4 as médias externas. O p-valor do teste X^2 para as médias apenas do
90 comando do Python foi de $p - \text{valor} = 5.0985243504669014^{-35}$, ou seja, um

91 valor extremamente abaixo de $p\text{-valor} = 0.05$ confirmando a diferença entre
92 as execuções internas e externas.

93 4.1. Projeto 2^{kr} fatorial com $k = 2$ e $r = 50$

94 Para verificar o impacto dos fatores no experimento, foi realizado um
95 projeto 2^{kr} fatorial com $k = 2$ e $r = 50$. Considerou-se o fator A como o
96 tamanho da amostra e o fator B como local de execução. A amostra de 1000
97 registro foi definida como -1 e a superior de 1. Como uma amostra pode
98 ser bem maior de um grupo de execução para o outro, foi feita a análise
99 utilizando a amostra superior primeiramente como 16.000 e depois a final de
100 1.024.000 registros. O fator B ficou com nível -1 quando a execução é externa
101 e 1 quando interna.

102 Projeto 1:

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

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

105

106 Projeto 2:

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

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

109 O resultado do Projeto 1 é apresentado na Tabela 3. Pode-se observar
110 que o fator considerando onde o comando foi executado explica 78.58% da
111 variação. O tamanho da amostra explica apenas 6.29%, menos que o a vari-
112 ação pelo erro de 8.82%. A estatística F calculada para esse projeto foi de
113 34.94 para os graus de liberdade 3 e 147, bem acima na tabela do valor de
114 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).

115 Já para o segundo projeto, como a distância entre a quantidade inicial
116 e final é bem mais relevante, os valores das proporções se alteraram signifi-
117 cativamente. A Tabela 4 apresenta os resultados. A estatística F para esse
118 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).

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

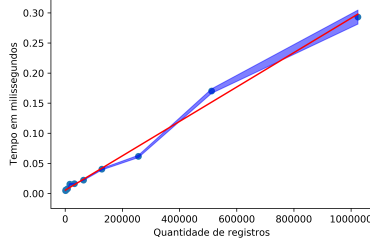
123 4.2. Análise gráfica e regressão

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

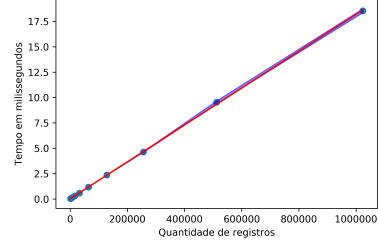
130 Os gráficos da Figura 1a e Figura 1c apresentam os resultados da execu-
131 ção apenas do comando do Python, no ambiente externo e interno respec-
132 tivamente. Como o *K-means* realiza o cálculo de distância entre todos os
133 pontos para cada *centroid* até a convergência, espera-se uma complexidade
134 seja $O(kn)$. Nesse experimento, k foi mantido constante, e assim é espe-
135 rado um crescimento linear no tempo de execução para o algoritmo isolado.
136 Observando o gráfico, percebe-se que existe, para os pontos medidos, uma
137 tendência de se manter próximo da reta de regressão. Considerando a apa-
138 rente aderência na execução total, pode-se supor que a taxa de transferência
139 dos dados entre o banco e a aplicação seja linear com relação ao número de
140 registros, entretanto o valor máximo da amostra pode se ainda baixo para
141 evidenciar uma tendência não linear.

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

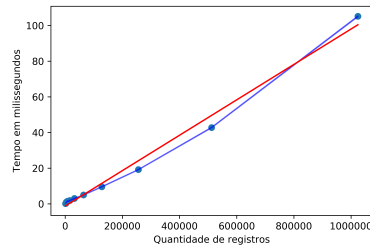
146 A Figura 1c e a Figura 1d apresentam os dados coletados na execução
147 interna com o tempo do Python e Total respectivamente. Diferente da exe-



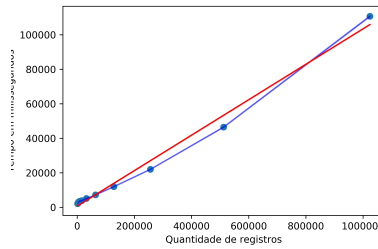
(a) Execução Externa apenas Python.



(b) Execução Externa Total.



(c) Execução Interna apenas Python.



(d) Execução Interna Total.

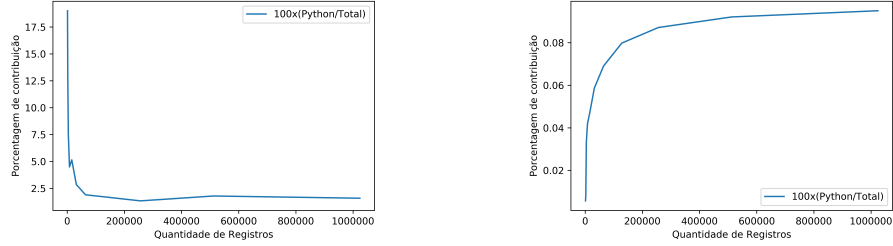
Figura 1: Evolução das médias com relação à quantidade de registros.

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

Regressão	Coefficientes	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.

153 Por fim, a Figura 2 apresenta a contribuição da execução do *K-means*
 154 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.

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

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

166 5. Conclusão

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

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

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

177 Embora que a execução interna ainda não tenha um desempenho supe-
 178 rior, ao menos para esse caso testado, a possibilidade de execução de ambi-

179 entes completos de linguagem de programação permite simplificar operações
180 por manter o dado próximo ao processamento e pode ser tornar uma ferra-
181 mente essencial futuramente.

182 [1] C. Yang, Q. Huang, Z. Li, K. Liu, F. Hu, Big data and cloud computing:
183 innovation opportunities and challenges, *International Journal of Digital*
184 *Earth* 10 (2017) 13–53.

185 [2] S. Makridakis, The forthcoming artificial intelligence (ai) revolution: Its
186 impact on society and firms, *Futures* 90 (2017) 46–60.

187 [3] R. A. Rodrigues, L. A. Lima Filho, G. S. Gonçalves, L. F. Mialaret, A. M.
188 da Cunha, L. A. V. Dias, Integrating nosql, relational database, and the
189 hadoop ecosystem in an interdisciplinary project involving big data and
190 credit card transactions, in: *Information Technology-New Generations*,
191 Springer, 2018, pp. 443–451.

192 [4] A. K. Jain, Data clustering: 50 years beyond k-means, *Pattern recogni-*
193 *tion letters* 31 (2010) 651–666.

194 [5] T. White, Hadoop: The definitive guide, "O'Reilly Media, Inc.", 2012.

195 [6] K. Shvachko, H. Kuang, S. Radia, R. Chansler, The hadoop distributed
196 file system, in: *Mass storage systems and technologies (MSST)*, 2010
197 IEEE 26th symposium on, Ieee, pp. 1–10.

198 [7] V. Mansinghka, R. Tibbetts, J. Baxter, P. Shafto, B. Eaves, Bayesdb: A
199 probabilistic programming system for querying the probable implications
200 of data, *arXiv preprint arXiv:1512.05006* (2015).