Compressão de imagens em uma arquitetura de *streaming* usando K-Means

Samuel Amico Fidelis

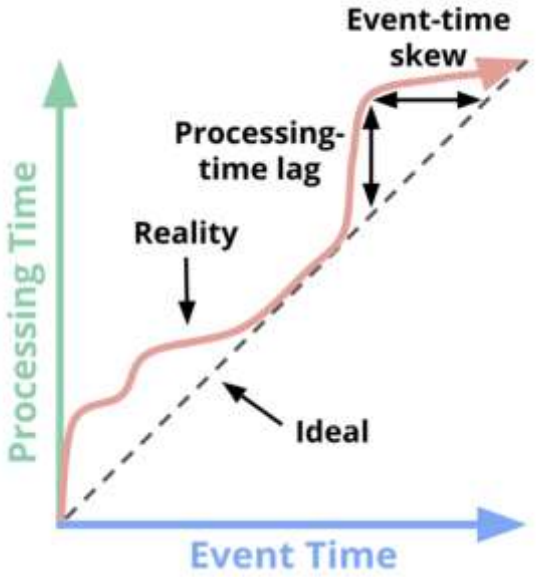
Programa de Pós Graduação em Ciência da Computação
Universidade Federal de Santa Catarina

Disciplina – Programação Paralela
Professor - Dr. Márcio Castro

Sumário

Introdução

- Processamento de dados em *streaming*
- Desafio: Domínio do tempo
  - Event Time – Processing Time
- Processamento em memória => Solução operacionalmente custosa



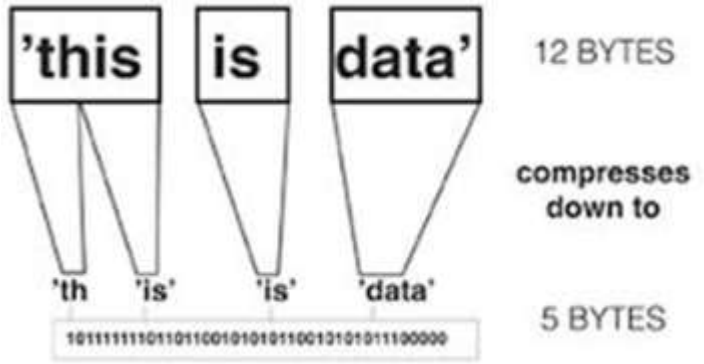
Introdução - Compressão

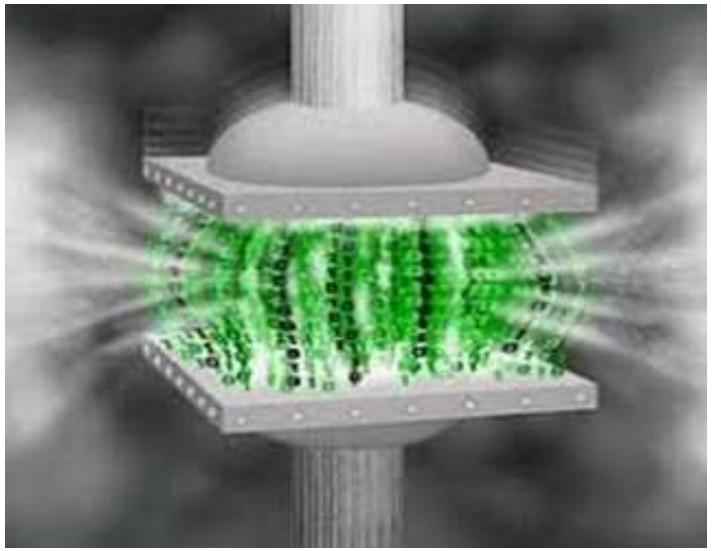
- Amplamente empregada para minimizar a utilização de disco, o consumo de largura de banda de rede e reduzir o consumo de energia do hardware.

- Compressão em *streaming* é distinta da compressão em Banco de Dados.

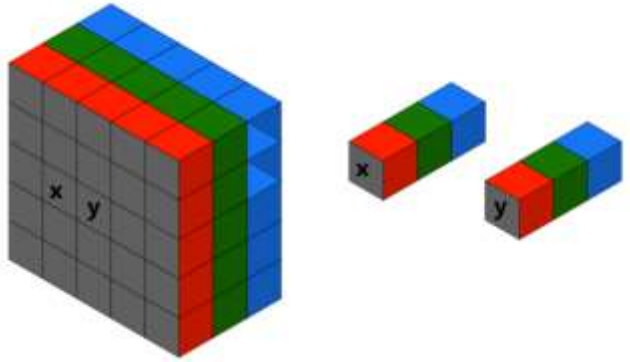
Motivação

- Compressão de dados ajuda a diminuir custos computacionais.
- Diminuir a taxa de E/L nos brokers de mensageria.
- Diminuir o tamanho dos dados para armazenamento *hot* nas filas de mensageria.





Motivação - Imagens

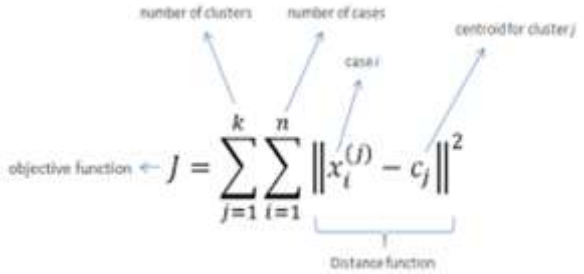


- Kafka não dispõe em suas propriedades de um tipo específico de compressão dedicada para imagens.

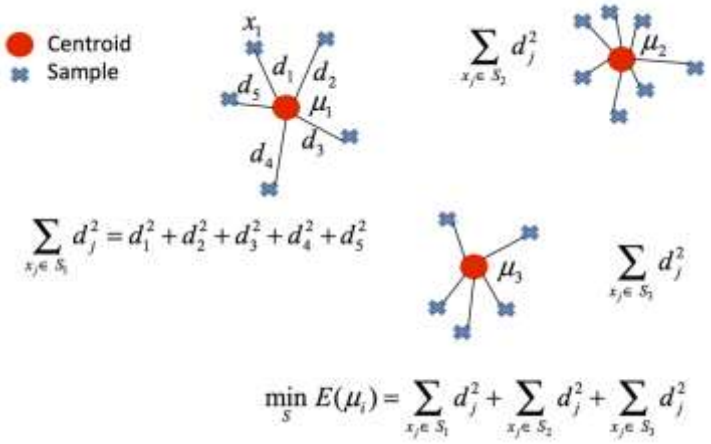
- Informações Irrelevantes.

Algoritmo

- K-Means – algoritmo de *clustering*



- Etapas:
  - Fornecer os valores para os centroides – os k centroides devem receber valores iniciais;
  - Gerar uma matriz de distância entre cada ponto e os centroides predefinidos;
  - Colocar cada ponto nas classes de acordo com sua distancia do centroide da classe;
  - Calcular os novos centroides para cada classes;
  - Repetir ate a convergência.



Solução

- K-Means – Multiprocessadores de memória compartilhada
- Etapas:
  - Fornecer os valores para os centroides – os k centroides devem receber valores iniciais;
  - Gerar uma matriz de distância entre cada ponto e os centroides predefinidos; (A associação de um ponto ao seu cluster é independente dos outros pontos, da mesma forma o cálculo da média de um cluster é independente dos outros);
  - Sincronização dos valores de média obtidos pelas threads – colocando cada ponto nas classes de acordo com sua distância do centroide. Calcular os novos centroides;
  - Repetir ate a convergência.

Solução

- Linguagem de Programação utilizada foi: Scala;
- Utilizei a abstração top-level de paralelismo nas coleções de pontos/matrizes;

Resultados

- Intel i7-6500 com 8 núcleos funcionando a 2,5 GHz e com 12 GB de RAM. Foram escolhidas um total de 40 imagens.
- Broker Kafka em um VM local.
- Foram realizados 6 testes com valores de K (numero de clusters) diferentes.
- Para cada teste foi realizado um conjunto de 60 trials aplicando o warmup do Java, remoção de outliers, média e cálculo de variância dos tempos de execução do algoritmo.

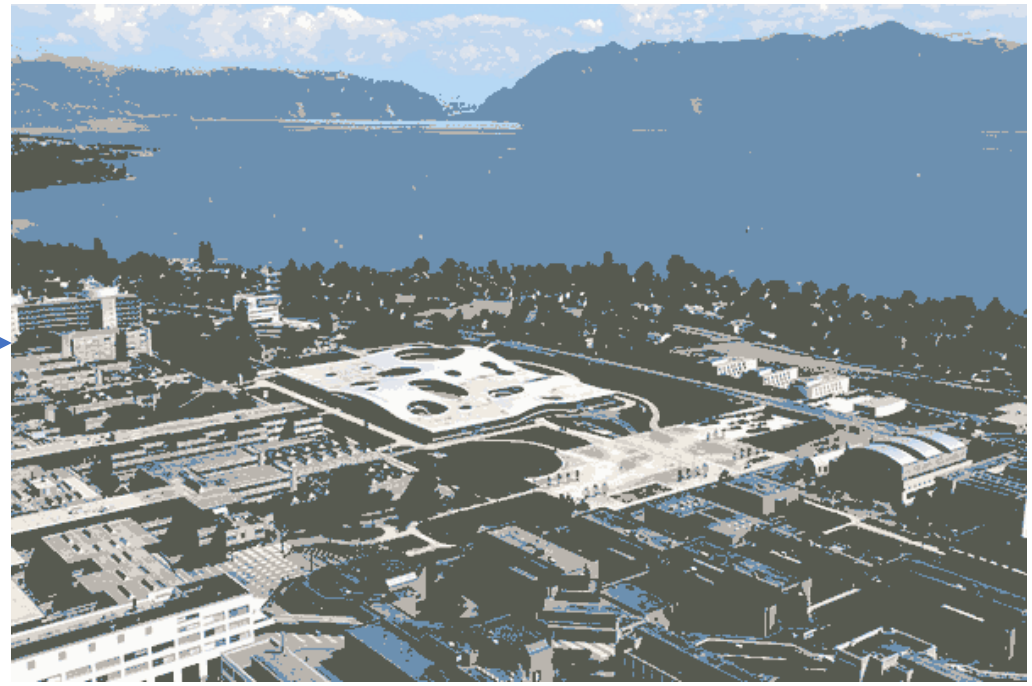
TEMPO DE EXECUÇÃO E TAMANHO DAS IMAGENS COMPRIMIDAS

K	Sequêncial(ms)	Paralelo(ms)	Tamanho (Kb)
12	36.407	19.3483	36
26	50.105	30.208	85
28	53.002	31.092	85
32	53.502	30.952	93
42	51.239	34.802	105
62	90.812	49.7142	117

USO DE CPU COM IMAGENS COMPRIMIDAS

K	CPU (%)	I/O (%)
12	1.5	2.00
26	2.1	2.40
28	2.1	2.42
32	2.5	2.47
42	2.7	2.61
62	3.33	3.09

K = 12

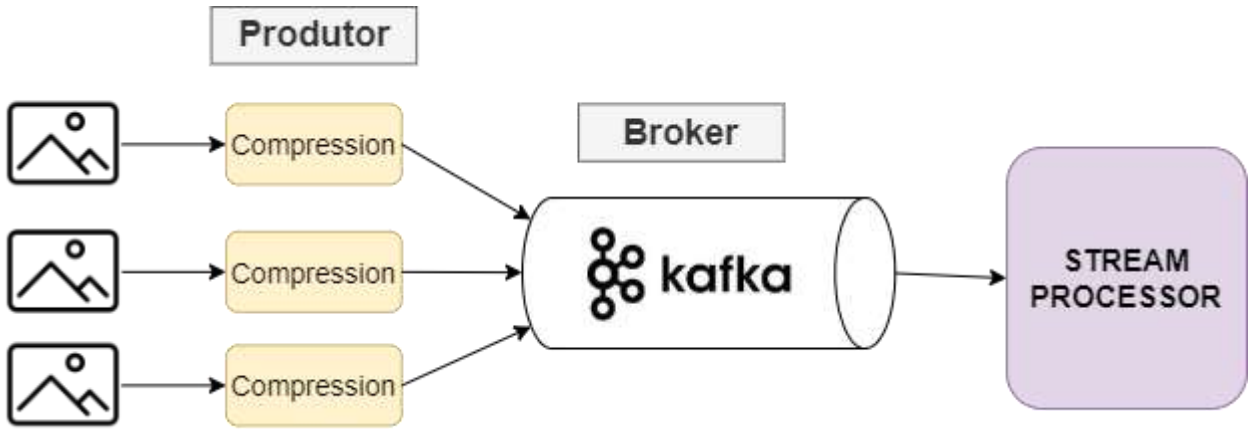


K = 28

K = 62

Resultados / Arquitetura

- Na arquitetura de processamento de streaming, a compressão de dados é realizada pelo produtor antes de enviar os eventos para o broker da fila de mensageria.



Conclusão

- O algoritmo do K-means apresentou um desempenho satisfatório ao diminuir o tamanho (em KB) das imagens processadas e sua versão paralela apresentou um tempo de execução igualmente satisfatório para uma arquitetura de streaming;

- Além disso, gerou a diminuição do uso de recursos de CPU e memória no broker. Vale ressaltar que os resultados foram obtidos com apenas uma maquina executando todo trabalho de processamento, ou seja, somente um nó responsável.

Referências

- [1] Akidau, Tyler, Slava Chernyak, and Reuven Lax. Streaming systems: the what, where, when, and how of large-scale data processing. ” O’Reilly Media, Inc.”, 2018.
- [2] Pekhimenko, Gennady, et al. ”Tersecades: Efficient data compression in stream processing.” 2018 USENIX Annual Technical Conference (USENIXATC 18). 2018.
- [3] Barga, Roger S., et al. ”Consistent streaming through time: A vision for event stream processing.” arXiv preprint cs/0612115 (2006).
- [4] SPARK, Apache. Apache spark. Retrieved January, v. 17, p. 2018, 2018.
- [5] Graefe, Goetz, and Leonard D. Shapiro. Data compression and database performance. University of Colorado, Boulder, Department of Computer Science, 1990.
- [6] Gonzalez, Rafael C., Richard Eugene Woods, and Steven L. Eddins. Digital image processing using MATLAB. Pearson Education India, 2004.
- [7] Banerjee, Ayan, and Amiya Halder. ”An efficient image compression algorithm for almost dual-color image based on k-means clustering, bitmap generation and RLE.” 2010 International Conference on Computer and Communication Technology (ICCCT). IEEE, 2010.
- [8] Dehariya, Vinod Kumar, Shailendra Kumar Shrivastava, and R. C. Jain. ”Clustering of image data set using k-means and fuzzy k-means algorithms.” 2010 International Conference on Computational Intelligence and Communication Networks. IEEE, 2010.
- [9] Pujari, Arun K. Data mining techniques. Universities press, 2001.
- [10] Alpaydin, Ethem. Introduction to machine learning. MIT press, 2020.
- [11] Jain, Anil K., and Richard C. Dubes. Algorithms for clustering data. Prentice-Hall, Inc., 1988.
- [12] Gersho, A., and R. M. Gray. ”Vector quantization and signal compression Boston.” Kluwer Academic (1992): 309-315.
- [13] Kucukyilmaz, Tayfun, and University of Turkish Aeronautical Association. ”Parallel k-means algorithm for shared memory multiprocessors.” Journal of Computer and Communications 2.11 (2014): 15.