Análise de Processamento Distribuído da Operação *Hash Join*[*]

Marisa S. Franco, Simone Dominico, Eduardo C. de Almeida, Marco A. Z. Alves

[1]Departamento de Informática – Universidade Federal do Paraná (UFPR)

{masf18,sdominico,eduardo,mazalves}@inf.ufpr.br

Resumo. *A Software-defined wide-area networking (SD-WAN) é uma tecnologia que permite (re)programar dispositivos de rede via software. Nesse caso, os dispositivos de rede possuem uma flexibilidade maior na administração, permitindo a programabilidade da rede. Este artigo apresenta uma avaliação do processamento distribuído de operadores hash join de banco de dados em dispositivos de rede, a partir de uma análise baseada em modelo de custo.*

1. Introdução

Quantidades massivas de dados têm sido geradas em taxas crescentes. Paralelamente, a capacidade computacional de coleta, processamento e armazenamento de dados aumentou exponencialmente na última década. Nesse contexto, as pesquisas sobre bancos de dados distribuídos ganharam cada vez mais relevância. O principal foco dos estudos é o custo de comunicação para transferência dos dados que sofrem, principalmente, com a latência de rede. Trata-se de um gargalo importante para melhoria de desempenho de consultas em sistemas de banco de dados nas quais a comunicação entre servidores é exigida, como aquelas nas quais são realizadas operações de junção. Além disso, com a expansão dos serviços na Internet, a tradicional *Wide Area Network* (WAN) perde espaço para uma nova tecnologia, a SD-WAN. Isso ocorre devido à baixa flexibilidade no uso da WAN tradicional para alguns acessos, como em serviços em *cloud*. A SD-WAN busca reduzir a complexidade da WAN por meio da virtualização de serviços.

Dessa forma, o roteador possui uma flexibilidade maior na administração, permitindo a programabilidade da rede via *softwares* desenvolvidos para esta finalidade [Shin et al. 2012]. Assim, elementos como os roteadores assumem uma posição de destaque devido a sua maior flexibilidade. A programabilidade da arquitetura SD-WAN proporciona novas possibilidades para mitigar latências de forma dinâmica, realizando o processamento das consultas nesses dispositivos. Na literatura, estudos do tipo *survey* tanto sobre operações de *join* [Yang and Singhal 1997] quanto sobre a tecnologia SD-WAN [Yang et al. 2019] não abordam o processamento de consultas em roteadores. Diante desse cenário, este artigo tem como objetivo avaliar o processamento distribuído de *hash join* em dispositivos de rede, por meio de uma análise baseada em modelo de custo. A análise realizada contempla a divisão da carga de trabalho de diferentes formas, incluindo o processamento total das consultas no roteador. Dessa forma, avaliou-se o potencial do processamento de dados em dispositivos de rede para ganho de desempenho.

2. Bancos de Dados Distribuídos e Operação de *Join*

Um sistema de banco de dados distribuído é uma coleção de bases de dados locais e remotos conectados através de uma rede de computadores. Utilizando vários computadores

[*]Este trabalho é parcialmente suportado pela CAPES e Instituto Serrapilheira (grant Serra-1709-16621).

chamados de nós, um banco de dados distribuído divide um problema grande em partes menores e menos complexas que são resolvidas de forma distribuída. As arquiteturas distribuídas de banco de dados são baseadas, principalmente, no modelo "cliente-servidor". Nessa arquitetura, o nó cliente envia uma consulta para nós servidores. Esses, por sua vez, respondem com os dados necessários para a consulta. No processamento de consultas distribuídas, a transferência de dados entre os nós de computação acontece por meio de uma rede dedicada ou de Internet. O custo destas consultas está diretamente ligado ao custo da transferência de dados na rede [Kossmann 2000]. Assim, diferentes algoritmos de processamento de consulta concentram-se na geração de planos de consulta que minimizem a quantidade de dados transferidos na rede. As consultas que possuem a operação de *join* utilizam *semi-joins* e *bloom filter* para minimizar a transferência dos dados na rede. Uma operação de *join* em uma consulta é um modo de recuperar dados de várias tabelas de banco de dados relacionais por meio de combinações lógicas entre as tabelas.

3. Metodologia e Modelagem Teórica

Para a avaliação do processamento distribuído de *hash join*, foram adotados alguns cenários de transmissão de dados baseados em topologia estrela, com dois servidores de banco de dados e um cliente conectados por um roteador. Foi escolhida ainda uma simplificação da consulta (*query-10*) do *benchmark* TPC-H [Council 2020], incluindo apenas a operação de *join*. A consulta foi implementada em C[1]. Para a criação das tabelas *hash*, foi utilizada a função FNV-1a [Fowler et al. 2013] – com excelente desempenho em trabalhos anteriores [Estébanez et al. 2014][Scheidt de Cristo et al. 2019]. Foram considerados seis cenários de processamento distribuído de *hash join* detalhados na Tabela 1.

Tabela 1. Cenários de processamento distribuído de *hash join* analisados

Armazenamento inicial	Processamento	Cenário
Relação maior no servidor 1 e menor no servidor 2.	Relação maior → Relação menor. Processamento no servidor 2.	1
	Relação menor → Relação maior. Processamento no servidor 1.	2
	Ambas → Roteador. Processamento no roteador.	3
Cada servidor possui uma metade das duas relações.	Cada servidor faz uma fase de análise. Troca de dados das tabelas *hash* entre servidores. Cada servidor processa parte da análise, usando as duas tabelas *hash* e sua metade da relação maior.	4
	Todas → Roteador. Processamento no roteador.	5
Cada servidor possui uma metade das duas relações, com dados relevantes co-localizados.	Processamento local.	6

Cada cenário foi avaliado com três cargas de trabalho (1 GB, 10 GB e 100 GB) e usando três tecnologias de rede (*Ethernet* 1 Gb, *Ethernet* 10 Gb e *InfiniBand* HDR 12×), com comunicação *half-duplex* e *full-duplex*. No caso dos cenários 3 e 5, foram avaliadas ainda três velocidades de processamento do roteador: igual às CPUs dos servidores, 5% e 10% mais lentas do que as CPUs. Nos experimentos, foi utilizada uma máquina com dois soquetes, cada um deles com um *Intel Xeon Silver* 4114 executando o sistema operacional Ubuntu, versão 18.04.01 LTS. Para obtenção dos tempos de execução para o cálculo, executou-se 20× cada cenário base, combinando as variáveis a seguir: tamanho da carga de trabalho (1 GB, 10 GB e 100 GB de armazenamento), capacidade de rede, paralelismo da rede, velocidade do roteador. Para os resultados, foram usadas as médias dos tempos das fases de construção, análise e impressão dos resultados em cada cenário base. A Tabela 2 descreve os cálculos feitos para os tempos de execução em cada cenário.

[1] *Hash join*: https://github.com/marisasel/hashjoin

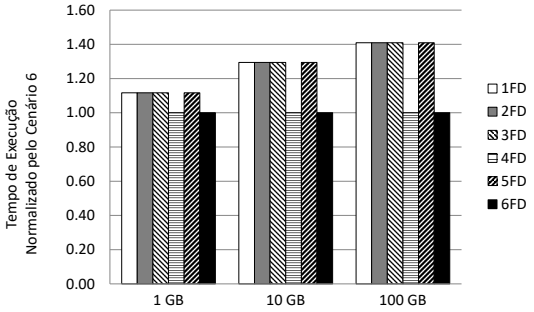
Tabela 2. Cálculo de custo utilizado em cada cenário

Cenário	Tempo de processamento da consulta	
	Half-duplex	***Full-duplex***
1	$TP + ((N + R)/BW)$	$TP + ((N + R)/BW)$
2	$TP + ((n + R)/BW)$	$TP + ((n + R)/BW)$
3	$TP + ((N + n + R)/BW)$	$TP + (((máx(N; n)) + R)/BW)$
4	$TP + ((H + R)/BW)$	$TP + (((H/2) + R)/BW)$
5	$TP + ((N + n + R)/BW)$	$TP + (((máx(N; n)) + R)/BW)$
6	$TP +$ (R / BW)	$TP +$ (R / BW)

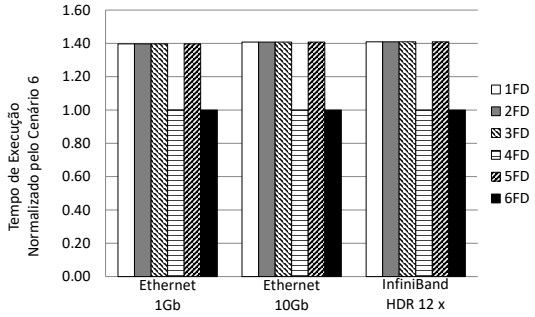
Obs.: Tempo de proc. nos cenários: 1, 2, 3 e 5: análise + construção + impressão de resultados. Tempo de proc. nos cenários 4 e 6: análise "paralela" + construção "paralela" + impressão de resultados. TP: Tempo de processamento. N: Tamanho da relação maior. n: Tamanho da relação menor. H: Tamanho da tabela *hash* - espalhamento perfeito. R: Tamanho da relação de saída. BW: Largura de banda. "Tamanho" refere-se à quantidade de tuplas da tabela multiplicada pela soma dos tamanhos dos tipos de dados usados para armazenar os conteúdos das colunas pertinentes à consulta.

4. Análise dos Resultados

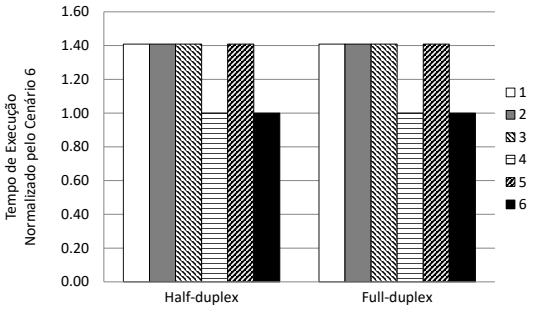
A Figura 1 mostra os resultados obtidos com a *query-10* do TPC-H por meio do cálculo dos tempos de processamento em cada cenário, detalhados na Tabela 2.



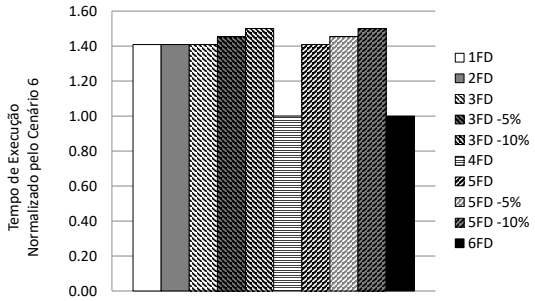
(a) InfiniBand HDR, *full-duplex*, roteador 100%.



(b) 100 GB, *full-duplex*, roteador 100%.



(c) 100 GB, InfiniBand HDR, roteador 100%.



(d) 100 GB, InfiniBand HDR, *full-duplex*.

Figura 1. Avaliação de desempenho da "*Query-10*", usando FNV-1a, com os tempos de execução normalizados pelo cenário 6.

A Figura 1 (a) mostra "melhor cenário": InfiniBand HDR 12×, *full-duplex*, roteador 100%. Na Figura 1 (b), a combinação dos parâmetros expressa as condições de "maior estresse à rede", enquanto a Figura 1 (c) apresenta o impacto da variação da capacidade de rede. Já a Figura 1 (d) mostra o impacto das diferentes velocidades de roteador. Em todos os experimentos, o cenário 6 apresentou o melhor desempenho em relação ao tempo de execução. No "melhor cenário" – Figura 1 (a) –, por exemplo, o cenário 6 teve tempo de execução de 6321 ms com carga de trabalho de 1 GB, de 83400 ms com 10 GB e de 1202534 ms com 100 GB. Tais resultados evidenciam a importância da transferência de dados na composição do custo total de processamento

distribuído de *hash join*. O cenário 6 é o que transfere a menor quantidade de dados: apenas os dados da relação de saída de consulta. Ao mesmo tempo, possui o maior sobrecusto de armazenamento e sincronização de dados nos servidores, pois os dados estão co-localizados. O segundo melhor desempenho foi obtido com o cenário 4, que trafega a segunda menor quantidade de dados: *hash join* e relação de saída. O cenário 1 deveria ser mais lento do que o 2, pois o primeiro transfere dados da relação maior e o segundo da menor. No entanto, ambos equiparam-se porque são solicitados mais dados provenientes da relação menor. O desempenho em relação ao tempo de execução com comunicação *full-duplex* dos cenários 3 e 5, quando com capacidade do roteador idêntica à das CPUs dos servidores, é igual ao dos cenários 1 e 2. Quando se adota a comunicação *half-duplex*, os cenários 1 e 2 são mais rápidos do que os cenários 3 e 5. À medida que se aumenta a velocidade da rede, diminui-se o *gap* entre os pares de cenários. Quanto menor a carga de trabalho, também é menor esse *gap*. Por fim, um processamento feito no roteador com capacidade 5% menor afeta o tempo total dos cenários 3 e 5 em 3,25%, enquanto um feito no roteador com capacidade 10% menor aumenta em 6,50%.

5. Conclusões e Trabalhos Futuros

Neste artigo, foi apresentada uma análise de diferentes cenários de execução de *hash join* distribuído, incluindo execução da operação em um roteador, a partir de um modelo de custo. Espera-se que os resultados aqui apresentados contribuam para um conhecimento mais amplo sobre diferentes estratégias de processamento distribuído de dados e o potencial do processamento de dados em dispositivos de rede para ganho de desempenho, motivando novos estudos. Como trabalhos futuros, pretende-se analisar outras consultas que realizam a operação de *hash join* com o modelo aqui apresentado e criar um ambiente virtualizado para simular os diferentes cenários e as implicações do processamento do *hash join* distribuído no roteador, simulando a tecnologia SD-WAN.

Referências

Council, T. P. (2020). Tpc benchmark h. `http://www.tpc.org/tpch/`. Acessado em 25/11/2020.

Estébanez, C., Sáez, Y., Recio, G., and Isasi, P. (2014). Performance of the most common non-cryptographic hash functions. *Software: Practice and Experience*, 44.

Fowler, G., Vo, P., and Noll, L. C. (2013). Fnv hash. `http://www.isthe.com/chongo/tech/comp/fnv/index.html/`. Acessado em 14/04/2020.

Kossmann, D. (2000). The state of the art in distributed query processing. 32(4):422–469.

Scheidt de Cristo, F., Almeida, E., and Alves, M. (2019). Vivid cuckoo hash: Fast cuckoo table building in simd. In *Simp. em Sistemas Computacionais de Alto Desempenho*.

Shin, M., Nam, K., and Kim, H. (2012). Software-defined networking (sdn): A reference architecture and open apis. In *Int. Conf. on ICT Convergence*, pages 360–361.

Yang, Y. and Singhal, M. (1997). A comprehensive survey of join techniques in relational databases.

Yang, Z., Cui, Y., Li, B., Liu, Y., and Xu, Y. (2019). Software-defined wide area network (sd-wan): Architecture, advances and opportunities. In *2019 28th International Conference on Computer Communication and Networks (ICCCN)*, pages 1–9.