Aumentando o Desempenho de Análises de Dados de Fluxos IP com Bancos de Dados Orientados a Coluna

Paulo Henrique de Oliveira[1], **Bruno Bogaz Zarpelão**[1], **Daniel S. Kaster**[1]

[1]Departamento de Computação – Universidade Estadual de Londrina (UEL)
Caixa Postal 10.011 – 86.057-970 – Londrina – PR – Brasil

oliveira.ph17@gmail.com, {brunozarpelao,dskaster}@uel.br

Abstract. *Data management has been one of the crucial problems in the development of new computing solutions, especially in this high-speed network age. In network traffic monitoring tasks, protocols such as NetFlow produce a huge amount of data which need to be handled efficiently. In that context, this work proposes employing column-oriented databases for boosting analyses over large IP flow databases. The paper shows experiments comparing row-oriented databases and column-oriented databases. The results achieved using the DBMSs PostgreSQL and MonetDB confirm the potential of column-oriented databases on that type of dataset.*

Resumo. *O gerenciamento de dados tem sido um dos problemas cruciais no desenvolvimento de novas soluções de computação, em especial na era das redes de alta velocidade. Em tarefas de monitoramento de tráfego de rede, protocolos como NetFlow produzem um volume gigantesco de dados que precisam ser manipulados eficientemente. Nesse contexto, este trabalho propõe o uso de bancos de dados orientados a coluna para aprimorar o desempenho de análises sobre fluxos IP. O artigo apresenta experimentos comparando bancos de dados orientados a linha e bancos de dados orientados a coluna. Os resultados obtidos usando os SGBDs PostgreSQL e MonetDB confirmam o potencial de bancos de dados orientados a coluna nesse tipo de conjunto de dados.*

1. Introdução

Os protocolos SNMP (*Simple Network Management Protocol*), NetFlow e IPFIX (*IP Flow Information Export*) são populares na área de gerência de redes devido ao fato de serem amplamente utilizados em tarefas de monitoramento [Deri et al. 2011]. Dentre esses protocolos, o SNMP é o mais antigo, tendo sido usado desde o final da década de 80 no monitoramento de tráfego em roteadores, através do qual são gerados dados estatísticos sobre a rede [Zhu et al. 2011]. No entanto, o SNMP não é capaz de fornecer informações de forma mais granular, como a quantidade de tráfego em diferentes tipos de aplicação. Nesse sentido, as informações de tráfego de rede fornecidas pelo NetFlow e pelo IPFIX são mais detalhadas.

O NetFlow, proposto em 2004, tem sido o protocolo mais utilizado para coletar informações de tráfego por meio da categorização de pacotes em fluxos IP e da obtenção de dados importantes referentes a esses fluxos, como endereços IP, portas TCP e UDP, contagens de pacotes e contagens de *bytes*. Porém, além de proporcionar uma maior qualidade de gerência, a riqueza de detalhes fornecidos pelo NetFlow traz o desafio de manipular eficientemente uma grande quantidade de dados, de forma a extrair informações

importantes dos mesmos. O gerenciamento de dados tem sido um dos problemas cruciais no desenvolvimento de novas soluções de computação, em especial na era das redes de alta velocidade [Balman and Byna 2011].

Existem inúmeras soluções de armazenamento e recuperação de dados disponíveis atualmente. Entretanto, grande parte das soluções existentes de apoio à gerência de redes utiliza implementações próprias para acesso aos dados de tráfego ou algum dos Sistemas de Gerenciamento de Bancos de Dados (SGBDs) mais conhecidos, tais como Oracle, PostgreSQL, MySQL e outros. Esses SGBDs têm em comum o armazenamento de dados orientado a linha, ou seja, são otimizados para a leitura/escrita de registros como um todo, que são formados por um conjunto de atributos que descrevem a entidade real armazenada no banco de dados. Porém, dependendo da operação de análise a ser realizada sobre os dados, apenas alguns atributos dos registros são necessários para produzir a resposta desejada. Esse tipo de análise é bastante frequente em monitoramento de redes usando fluxos IP. Nesse caso, em SGBDs com armazenamento orientado a linha, todos os atributos dos registros precisam ser recuperados, exigindo um esforço computacional desnecessário.

Outra técnica utilizada por alguns SGBDs é o armazenamento de dados orientado a coluna. Nessa abordagem, os valores de um mesmo atributo dos vários registros são armazenados de forma contígua, otimizando o acesso a um número reduzido de atributos de um conjunto consideravelmente grande de registros. Dessa forma, são soluções mais adequadas para a realização de análises de dados envolvendo um número restrito de atributos sobre um grande conjunto de dados, tendo sido utilizadas em *data warehouses*. Pela natureza dos dados de fluxos IP e o volume gigantesco de fluxos que podem ser gerados por redes de médio e grande porte, soluções de armazenamento orientadas a coluna podem ser mais eficientes que outras abordagens para suportar a análise desses dados. Segundo o nosso conhecimento, não há trabalhos que tenham feito uma comparação de desempenho de SGBDs orientados a linha e de SGBDs orientados a coluna para responder a consultas típicas sobre conjuntos de dados de fluxos IP e esse é o foco deste trabalho.

O objetivo deste trabalho é explorar o potencial de bancos de dados orientados a coluna para aprimorar o desempenho de análises de grandes volumes de dados de fluxos IP. O artigo descreve uma análise de como um banco de dados orientado a coluna pode ser utilizado para manipular fluxos IP em tarefas usuais de gerência de redes e compara o desempenho dessa abordagem ao desempenho de um banco de dados orientado a linha. A metodologia de análise adotada incluiu experimentos sobre um conjunto real de fluxos IP de acesso público, cujo tamanho excede 14 milhões de registros. Tais experimentos compreenderam a execução de um conjunto de consultas típicas sobre esse tipo de conjunto de dados, elencadas por trazerem informações importantes para o gerente de redes. Para executar cada um dos experimentos, foram utilizados os SGBDs PostgreSQL, cujo armazenamento é orientado a linha, e MonetDB, cujo armazenamento é orientado a coluna. O MonetDB apresentou desempenho superior ao PostgreSQL para a maioria das consultas, chegando a ser até mais de 8 vezes mais rápido em alguns casos, o que confirma o potencial de soluções orientadas a coluna em tarefas de análise de fluxos de redes.

Este artigo está organizado da seguinte forma: a Seção 2 descreve os conceitos fundamentais ao entendimento do trabalho; a Seção 3 apresenta a proposta de utilização de bancos de dados orientados a coluna para análise de fluxos IP; a Seção 4 mostra os experimentos realizados para avaliar a proposta; e a Seção 5 apresenta as conclusões.

2. Fundamentação Teórica

2.1. Análise de Tráfego de Redes com o Protocolo NetFlow

O protocolo NetFlow foi proposto pela Cisco [Claise 2004]. Seu objetivo era prover ao administrador de redes um detalhamento maior sobre o funcionamento das redes IP do que o protocolo SNMP podia oferecer. De forma paralela, a organização de padrões abertos IETF (*Internet Engineering Task Force*) propôs o IPFIX [Claise et al. 2013], protocolo cujo propósito é o mesmo do NetFlow. Segundo [Clemm 2006], a diferença mais importante entre ambos os protocolos é política: enquanto o IPFIX é apoiado por uma organização de padrões abertos, o NetFlow é um padrão amplamente consolidado, mas cuja especificação pertence à Cisco. Dessa forma, esses dois protocolos têm sido ferramentas fundamentais para analistas de rede nos últimos anos, com destaque para o NetFlow pela maior disseminação.

O NetFlow comunica informações estatísticas de fluxos IP. Um fluxo consiste em todo tráfego pertencente ao mesmo contexto de comunicação. Cada fluxo IP é unicamente identificado pelas chaves: endereço IP de origem, porta de origem, endereço IP de destino, porta de destino, tipo do protocolo (TCP ou UDP) e tipo do serviço (campo TOS). Cada registro de fluxo inclui, além das chaves, os tempos de início e fim do fluxo e a quantidade de pacotes e *bytes* transportados. Um pacote NetFlow é formado por um cabeçalho e por uma sequência de registros de fluxo. O cabeçalho contém o número de sequência do pacote, a quantidade de registros de fluxo e o número da versão do protocolo NetFlow. Os registros de fluxo compreendem as chaves, que os identificam unicamente, e seus dados estatísticos.

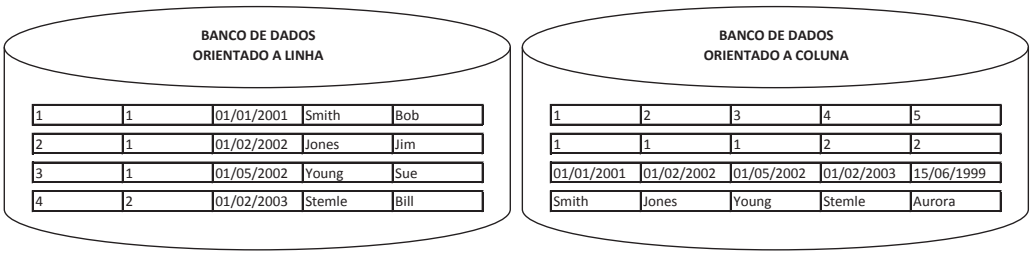
Os dados sobre o tráfego da rede coletados pelo protocolo NetFlow permitem responder diversas questões [Clemm 2006]. Esses dados são particularmente úteis para aplicações de gerência que oferecem funcionalidades voltadas a contabilidade e a gerenciamento de desempenho. Algumas consultas típicas que podem ser respondidas com base nessas informações são: (i) quais são os maiores *talkers* da rede? (ii) qual é a quantidade de tráfego entre dois pontos específicos da rede? (iii) como os enlaces da rede estão sendo utilizados? (iv) onde estão os gargalos na rede? Pode-se observar que, para responder a muitas dessas consultas, faz-se necessário agregar dados de um conjunto potencialmente grande de fluxos IP. Dado que o volume de dados de tráfego gerados em uma rede de médio ou de grande porte é grande, os repositórios de dados crescem rapidamente e continuamente, demandando soluções escaláveis para armazenamento e recuperação de dados. Uma alternativa pouco explorada para otimizar o desempenho de análises sobre grandes conjuntos de fluxos IP é o uso de bancos de dados orientados a coluna, introduzidos a seguir.

2.2. Bancos de Dados Orientados a Coluna

A maioria dos SGBDs tradicionais utiliza arquiteturas de armazenamento orientadas a linha, em que todos os atributos de um registro são gravados contiguamente em disco [Stonebraker et al. 2005]. Com essa arquitetura, uma única operação de escrita no disco é suficiente para armazenar todos os atributos de um registro, o que torna possível obter um bom desempenho em operações de escrita de um registro inteiro. Considando o conjunto de dados de exemplo apresentado na Figura 1, um SGBD orientado a linha armazena os dados da forma apresentada na Figura 2(a).

empregado_id	departamento_id	data_contrato	sobrenome	nome
1	1	01/01/2001	Smith	Bob
2	1	01/02/2002	Jones	Jim
3	1	01/05/2002	Young	Sue
4	2	01/02/2003	Stemle	Bill
5	2	15/06/1999	Aurora	Jack
6	3	15/08/2000	Jung	Laura

Figura 1. Conjunto de dados de exemplo.



(a) Armazenamento orientado a linha. (b) Armazenamento orientado a coluna.

Figura 2. Comparação entre o armazenamento orientado a linha e a coluna.

Nos últimos anos, a área de gerenciamento de dados tem visto um crescimento no interesse pelo armazenamento de dados orientado a coluna no projeto de sistemas de bancos de dados otimizados para leitura [Min and Franke 2010]. Consultas analíticas de negócio e *data warehouses* têm sido os principais alvos desse tipo de armazenamento. Em SGBDs orientados a coluna, o armazenamento de uma tabela se dá pelo seu particionamento em arquivos de coluna única [Harizopoulos et al. 2006]. Cada arquivo de coluna única armazena os valores de sua respectiva coluna na tabela. Com isso, todos os valores de uma mesma coluna podem ser acessados de forma consecutiva, diferentemente dos bancos de dados orientados a linha, em que os registros como um todo são armazenados contiguamente [Liu et al. 2011]. Considerando o conjunto de dados de exemplo da Figura 1, um SGBD orientado a coluna armazena os dados como ilustrado na Figura 2(b).

Com essa estrutura de armazenamento orientada a coluna, melhora-se o desempenho de consultas cujo objetivo é manipular colunas dos registros, isto é, valores de atributos específicos dos registros do conjunto de dados [Liu and Zhou 2013]. Esse cenário é comum em consultas analíticas, como no processamento analítico *online* (OLAP), *data mining* em larga escala e análises de inteligência de negócio em *big data*. Observando-se que as tarefas de análise de fluxos de rede fazem uso frequente de consultas analíticas, é possível vislumbrar que existe margem para aprimoramento de desempenho utilizando um banco de dados orientado a coluna. A próxima seção apresenta a proposta de avaliação de desempenho de bancos de dados orientados a coluna para esses tipos de análises.

3. Análise de Fluxos IP Utilizando Bancos de Dados Orientados a Coluna

Existem inúmeras consultas que podem ser realizadas sobre um banco de dados de fluxos IP. Considere-se, por exemplo, a base de dados construída no trabalho de [Sperotto et al. 2009]. No cenário de coleta dessa base de dados, um *honeypot*, executando a partir de uma máquina virtual, atuou durante 6 dias, do dia 23 de setembro de 2008 às 12:40:00 GMT ao dia 29 de setembro de 2008 às 22:40:00 GMT. O *honeypot* estava hospedado na rede da Universidade de Twente e diretamente conectado à Internet. A coleta de dados resultou em um arquivo de 24 GB contendo 155.2 milhões de pacotes,

que, após serem processados, geraram 14.2 milhões de fluxos e 7.6 milhões de alertas. Embora trate-se de um *honeypot* preparado para atrair tráfego, é possível perceber que o volume de bases de dados de fluxos tende a ser muito grande em ambientes reais. Os fluxos armazenados nessa base estão armazenados na tabela *flows*, apresentada na Figura 3.

id [PK] integer	src_ip bigint	dst_ip bigint	packets smallint	octets integer	start_time integer	start_msec smallint	end_time integer	end_msec smallint	src_port integer	dst_port integer	tcp_flags smallint	prot smallint
1	2463760020	3752951033	1	60	1222173605	985	1222173605	985	4534	22	2	6
2	2463760020	3752951032	1	60	1222173605	985	1222173605	985	1923	22	2	6
3	2463760020	3752951030	1	60	1222173605	985	1222173605	985	3185	22	2	6
4	2463760020	3752951031	1	60	1222173605	985	1222173605	985	2466	22	2	6
5	2463760020	3752951029	1	60	1222173605	985	1222173605	985	3056	22	2	6
6	2463760020	3752951028	1	60	1222173605	985	1222173605	985	4735	22	2	6
7	2463760020	3752951025	1	60	1222173605	985	1222173605	985	3210	22	2	6
8	2463760020	3752951026	1	60	1222173605	985	1222173605	985	2902	22	2	6
9	2463760020	3752951288	1	60	1222173605	985	1222173605	985	4509	22	2	6
10	2463760020	3752951289	1	60	1222173605	985	1222173605	985	3489	22	2	6

Figura 3. Primeiros 10 registros da tabela *flows*.

Se analisados isoladamente, os dados de fluxos não trazem informação relevante. Entretanto, se forem devidamente agregados, podem relevar padrões de tráfego extremamente úteis. Exemplos de consultas de grande interesse para gerentes de rede são apresentados a seguir. **Q1**: *quais são os 10 usuários que mais enviaram pacotes ou bytes?* **Q2**: *quais são os 10 serviços que mais demandaram pacotes ou bytes?* **Q3**: *quais são as 10 duplas de usuários que mais trocaram pacotes ou bytes entre si?* Considerando os atributos dos fluxos IP, os usuários dizem respeito aos endereços IP de origem, os serviços às portas de destino e as duplas de usuários aos pares de endereços IP de origem e de destino. Tais consultas, tipicamente, são executadas agregando resultados por intervalos de tempo, como: a cada intervalo de 5 minutos, a cada intervalo de 1 hora, a cada intervalo de 1 dia e durante todo o período.

O fato de armazenar os dados de fluxo em um banco de dados (orientado a linha ou a coluna) possibilita utilizar instruções SQL (*Structured Query Language*) para implementar essas e outras consultas. As consultas **Q1** e **Q2** podem ser representadas em SQL usando-se uma construção semelhante, conforme segue:

```
SELECT * FROM (
  SELECT X, Y, I, ROW_NUMBER() OVER (PARTITION BY I ORDER BY Y DESC) AS rn
  FROM (SELECT X, SUM(Z) AS Y, (start_time - 1222173605) / T AS I
        FROM flows WHERE dst_ip = 2463760020
        GROUP BY X, I
  ) subtable1
) subtable2 WHERE rn <= 10;
```

onde:

- X: atributo desejado (endereço IP de origem, porta de destino ou par de endereços IP de origem e de destino);
- Y: nome dado ao atributo que representa a soma de pacotes ou de *bytes*;
- I: intervalo de tempo considerado;
- Z: atributo referente a pacotes ou a *bytes* ($packets$ ou $octets$);
- T: valor que divide o tempo total nos intervalos desejados (300, 3600 ou 86400 segundos, isto é, 5 minutos, 1 hora ou 1 dia);
- 1222173605: valor mínimo do atributo $start\_time$;
- 2463760020: endereço IP anonimizado do *honeypot*.

Observe-se que o uso de funções analíticas de janela deslizante (*window functions*), definidas no padrão SQL, permite representar as consultas, que têm uma estrutura relativamente complexa, de forma simples e elegante. As mesmas consultas **Q1** e **Q2**, considerando-se o período todo, podem ser representadas de forma ainda mais simples:

```
SELECT X, SUM(Z) AS Y
FROM flows WHERE dst_ip = 2463760020
GROUP BY X ORDER BY Y DESC
LIMIT 10;
```

No caso da consulta **Q3**, que recupera as 10 maiores duplas de usuários por pacotes ou por *bytes*, pode-se também utilizar instruções SQL semelhantes, mas removendo-se a condição `WHERE dst_ip = 2463760020`, quando não se deseja considerar somente os fluxos cujo endereço IP de destino seja o do *honeypot*. É importante ressaltar também que as consultas apresentadas até aqui exigem agregar uma quantidade relativamente grande de fluxos para produzir os resultados desejados. Para isso, normalmente, faz-se uma varredura sequencial no conjunto de dados.

Outra classe de consultas de interesse consiste em efetuar análises considerando intervalos pontuais (e.g. nos últimos 5 minutos, na última hora ou em um dia específico), agregando uma quantidade reduzida de fluxos, tais como as consultas a seguir. **Q4**: *quais são os 10 usuários que mais enviaram pacotes ou bytes na última hora*? **Q5**: *quais são as 10 duplas de usuários que mais trocaram pacotes ou bytes entre si no dia 31/12/2013?* As consultas **Q4** e **Q5** podem ser expressas em SQL da seguinte forma:

```
SELECT X, SUM(Z) AS Y
FROM flows WHERE dst_ip = 2463760020 AND start_time BETWEEN T1 AND T2
GROUP BY X ORDER BY Y DESC
LIMIT 10;
```

onde as variáveis T1 e T2 representam o tempo de início e de fim considerados e as outras variáveis têm os mesmos significados descritos anteriormente. Uma característica importante desse tipo de consultas é que o fato de agregarem uma quantidade reduzida de fluxos potencializa o uso de índices para recuperar de forma eficiente os dados necessários.

Além das consultas apresentadas nesta seção, várias outras consultas relevantes para administradores de rede poderiam ser enumeradas. O ponto fundamental a ser observado é que a maior parte das consultas que realizam análises e identificação de padrões de interesse envolvendo dados de fluxos IP agrega uma parcela significativa de dados. Com base nessa observação, propomos a utilização de bancos de dados orientados a coluna para aumentar o desempenho dessas consultas e permitir a realização de análises complexas sobre grandes volumes de dados de forma escalável. A próxima seção apresenta experimentos realizados para avaliar nossa proposta.

4. Experimentos

Os experimentos consistiram na execução das consultas apresentadas na seção anterior na base de dados de fluxos IP de [Sperotto et al. 2009]. Para isso, foram utilizados dois SGBDs: PostgreSQL, cujo armazenamento é orientado a linha, e MonetDB, cujo armazenamento é orientado a coluna. Tais experimentos tiveram como foco avaliar o desempenho de ambos os SGBDs em termos de tempo de execução para cada consulta. O critério utilizado para a escolha das consultas foi baseado na importância das informações que elas

trazem para o gerente de redes. O PostgreSQL foi escolhido pelo fato de ser um SGBD orientado a linha amplamente utilizado em aplicações de propósito geral. O MonetDB, por sua vez, foi escolhido por ser um SGBD orientado a coluna que implementa grande parte das funcionalidades que o PostgreSQL possui, como funções de janela. Essa decisão de escolha dos SGBDs foi importante para que, por meio da execução dos experimentos, dois tipos de solução de armazenamento pudessem ser devidamente comparados.

A máquina utilizada nos experimentos possui as seguintes configurações: processador Intel Core i7 3612QM (6 MB de *cache*, 2.1–3.1 GHz), memória RAM de 8 GB *Dual Channel* DDR3 1600 MHz, disco rígido de 1 TB SATA 5400 RPM e sistema operacional Ubuntu 13.10, versão estável. Todas as consultas foram implementadas por meio da linguagem SQL. Cada consulta e inserção de dados foram executadas 10 vezes. Para cada uma delas, calculou-se a média e o desvio padrão dos 10 tempos de execução. Antes da execução de cada consulta e de cada inserção de dados, foram esvaziados os *caches* do sistema operacional e dos SGBDs. Nas próximas subseções, são detalhados os experimentos realizados seguidos de discussões dos resultados.

4.1. Resultados e Discussão

O primeiro experimento realizado teve como objetivo avaliar o desempenho dos SGBDs para responder às consultas **Q1**, **Q2** e **Q3**. Como pode ser visto nas Figuras 4, 5 e 6, o MonetDB apresentou um melhor desempenho em todos os intervalos considerados. Em cada gráfico desta subseção, é apresentado o tempo médio de execução juntamente com o desvio padrão para cada consulta executada 10 vezes. O desvio padrão é representado por uma barra preta de fina espessura na extremidade de cada barra maior. O ganho do MonetDB sobre o PostgreSQL chegou a ser até mais de 8 vezes maior nas consultas que retornam as 10 maiores duplas de usuários por *bytes* a cada 5 minutos e a cada 1 hora. Nota-se também que os tempos de execução do MonetDB foram parecidos para todos os intervalos. Considerando apenas o PostgreSQL, os tempos de execução das consultas para cada intervalo de 5 minutos, 1 hora e 1 dia foram estáveis. Porém, elas foram mais lentas do que as consultas que consideram todo o período. Isso acontece devido à manipulação dos dados através da função de janela $row\_number()$, utilizada para elencar os 10 maiores registros para cada um dos intervalos de tempo existentes no conjunto de dados. No caso das consultas que consideram todo o período, esse processamento não existe, o que torna mais rápida sua execução.

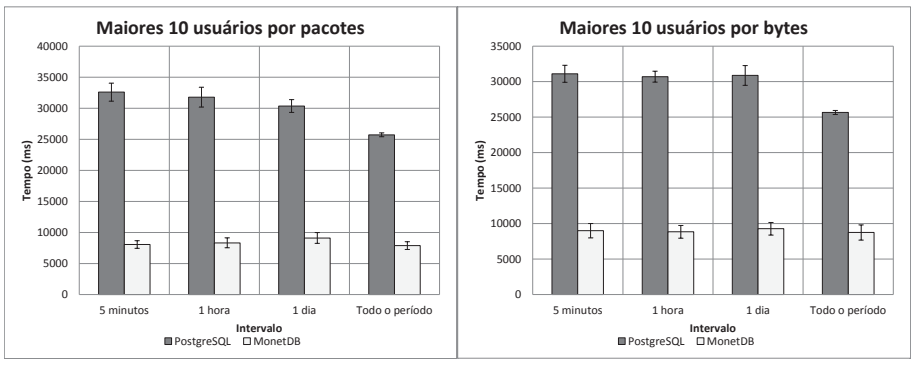


Figura 4. Maiores 10 usuários por pacotes e por *bytes*.

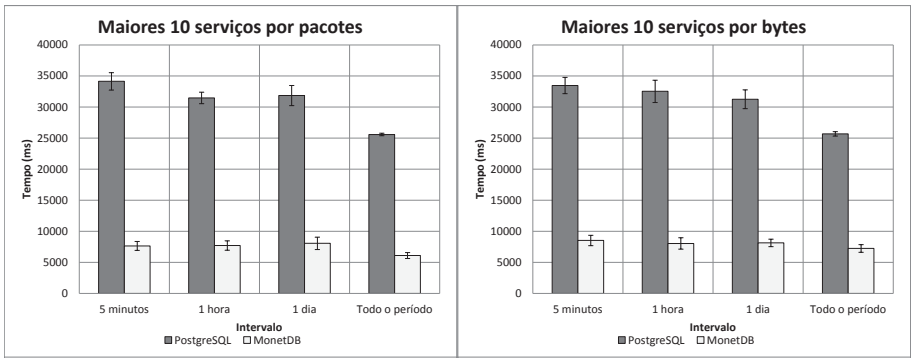


Figura 5. Maiores 10 serviços por pacotes e por *bytes*.

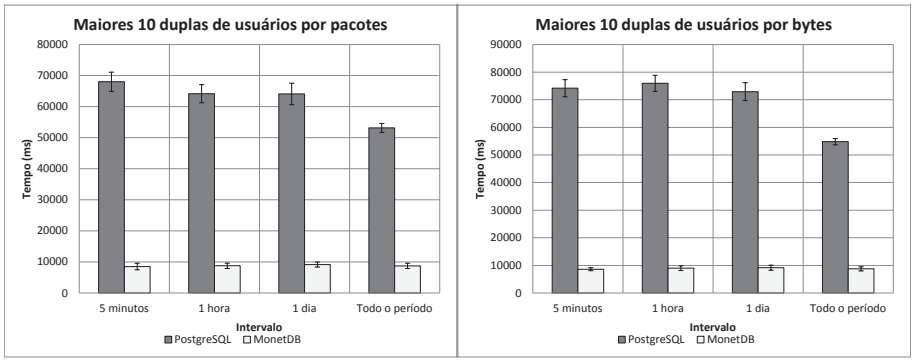


Figura 6. Maiores 10 duplas de usuários por pacotes e por *bytes*.

No segundo experimento, foram realizadas as mesmas consultas do experimento anterior, mas com o objetivo de verificar o desempenho de cada consulta em instâncias da tabela *flows* com diferentes números de registros: 1 milhão, 5 milhões, 10 milhões e 14 milhões. Além disso, ao invés de considerar todos os intervalos de tempo, foram considerados apenas os intervalos de 1 hora, pelo fato de serem utilizados com mais frequência. Os resultados obtidos são apresentados nas Figuras 7, 8 e 9. Como era esperado, quanto maior foi o tamanho da instância da tabela *flows*, mais tempo as consultas levaram para terminar a execução. Entretanto, em todas as consultas, o MonetDB apresentou um melhor desempenho, chegando a ser mais de 7 vezes mais rápido do que o PostgreSQL. Além disso, nota-se que, à medida que a tabela cresce, maior se torna a diferença entre o desempenho de ambos os SGBDs.

No terceiro experimento, a ideia central foi executar as consultas em intervalos pontuais **Q4** e **Q5**. Foi considerado um único intervalo de 5 minutos, 1 hora e 1 dia em instâncias da tabela *flows* com 1 milhão, 5 milhões, 10 milhões e 14 milhões de registros. O diferencial desse experimento em relação aos anteriores é que as consultas foram executadas de três formas: no PostgreSQL sem índice no atributo $start\_time$, no MonetDB e no PostgreSQL com índice no atributo $start\_time$. Não foi considerado o uso de índice no MonetDB pelo fato de ele próprio já tomar decisões acerca da necessidade do uso de índices. Como pode ser visto nas Figuras 10, 11 e 12, o MonetDB apresentou um melhor desempenho em algumas consultas e o PostgreSQL com índice no atributo $start\_time$ apresentou um melhor desempenho em outras consultas. No caso das consultas que recuperam os 10 maiores usuários por pacotes em 5 minutos e em 1 hora, o PostgreSQL com

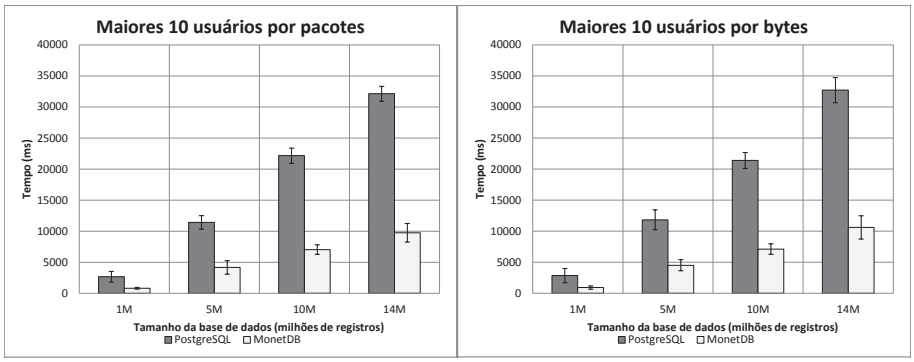


Figura 7. Maiores 10 usuários por pacotes e por *bytes* a cada 1 hora.

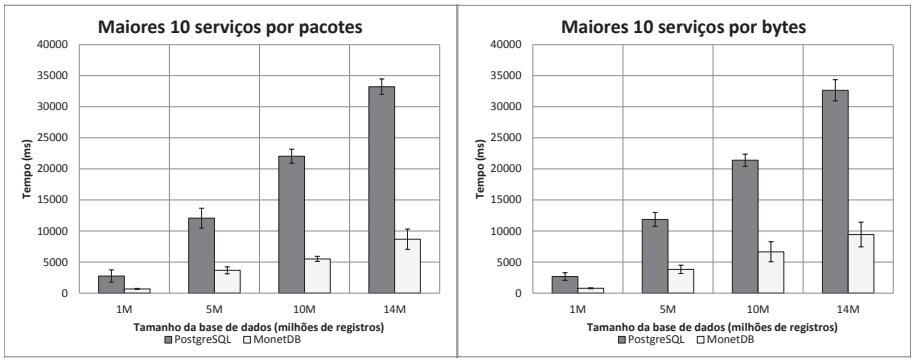


Figura 8. Maiores 10 serviços por pacotes e por *bytes* a cada 1 hora.

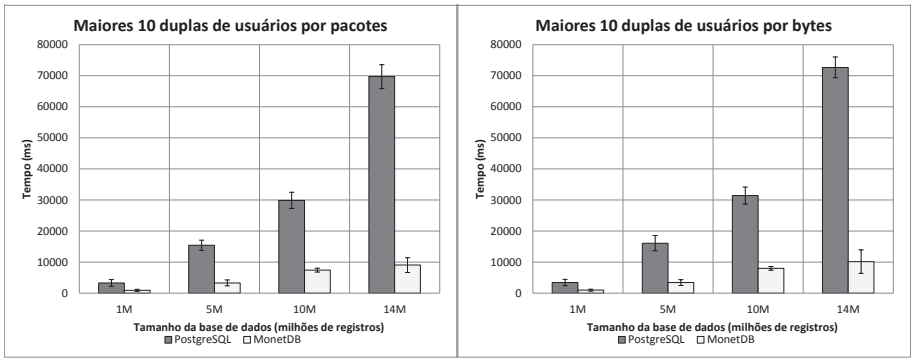


Figura 9. Maiores 10 duplas de usuários por pacotes e por *bytes* a cada 1 hora.

índice foi melhor do que o MonetDB de um modo geral. Já na consulta cujo intervalo de tempo é 1 dia, o MonetDB teve um desempenho melhor para todas as instâncias da tabela *flows*, chegando a ser até mais de 3 vezes mais rápido. Nesse caso, o PostgreSQL não fez uso do índice em $start\_time$ na instância de 1 milhão de registros. Isso aconteceu pelo fato de a seletividade da consulta ter sido menor, isto é, a quantidade de registros retornados foi maior, a ponto de não valer a pena o uso do índice. De 1 milhão de registros, 443498 foram retornados, o que equivale a 44% da tabela. Por esse motivo, seu tempo de execução foi praticamente o mesmo obtido com o PostgreSQL sem índice. No caso das consultas que recuperam as 10 maiores duplas de usuários por pacotes em 5 minutos e em 1 hora, o PostgreSQL com índice foi melhor do que o MonetDB para a maioria das instâncias da tabela *flows*. De um modo geral, os tempos de execução do PostgreSQL

com índice e do MonetDB foram parecidos. Já na consulta cujo intervalo de tempo é 1 dia, o MonetDB teve um desempenho melhor para todas as instâncias da tabela *flows*, sendo até mais de 5 vezes mais rápido do que o PostgreSQL. Na instância de 1 milhão de registros, o PostgreSQL não fez uso do índice em $start\_time$ pelo mesmo motivo citado anteriormente, visto que, nesse caso, 100% dos registros foram retornados.

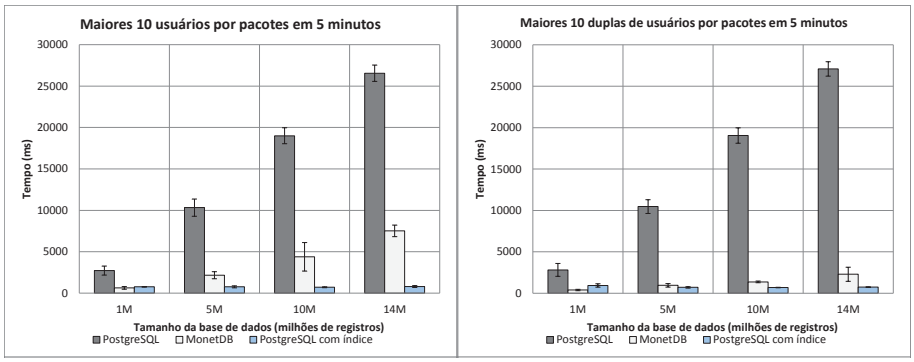


Figura 10. Maiores 10 usuários e duplas de usuários por pacotes em 5 minutos.

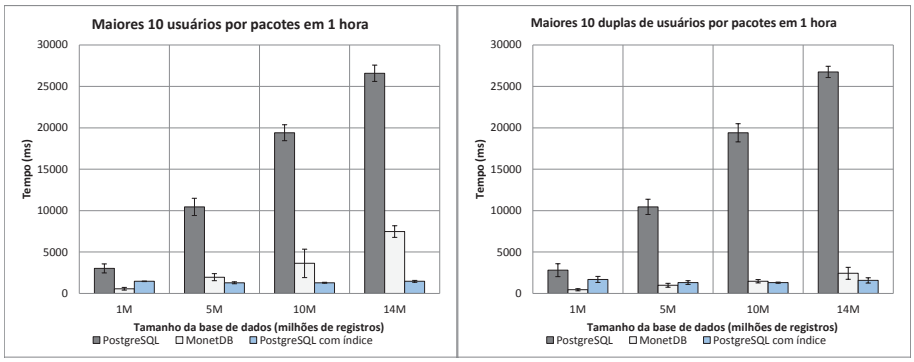


Figura 11. Maiores 10 usuários e duplas de usuários por pacotes em 1 hora.

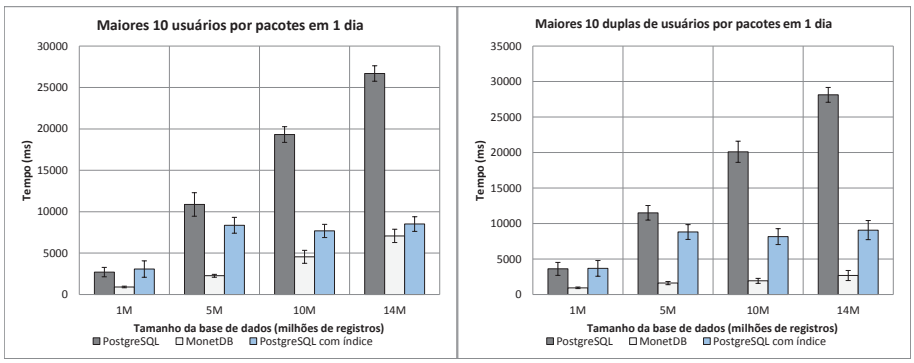


Figura 12. Maiores 10 usuários e duplas de usuários por pacotes em 1 dia.

No quarto experimento, ao invés de consultas de histórico, foram feitas operações de inserção/carga nos SGBDs considerando as instâncias de tamanhos diferentes da tabela *flows*. O objetivo foi verificar o desempenho do PostgreSQL sem índice em $start\_time$, do MonetDB e do PostgreSQL com índice em $start\_time$ na inserção de 500 registros.

Os registros foram inseridos por meio de operações de inserção comuns e por meio de *bulk loading*, nome dado a mecanismos de carga rápida que a maioria dos SGBDs implementa. Tais mecanismos são utilizados, especialmente, quando se deseja inserir uma grande quantidade de dados de uma só vez. Como pode ser observado na Figura 13, os tempos de inserção para o PostgreSQL sem índice em $start\_time$ e com índice em $start\_time$ foram parecidos, tanto através de operações de inserção comuns quanto através de *bulk loading*. No caso do PostgreSQL com índice em $start\_time$, os tempos de inserção foram levemente maiores, o que já era esperado, devido ao fato de o índice precisar ser atualizado conforme os registros são inseridos. Quanto ao MonetDB, sua grande desvantagem foi nesse experimento. Considerando apenas operações comuns de inserção, o MonetDB foi de 3 a 37 vezes, aproximadamente, mais lento do que o PostgreSQL. Considerando apenas *bulk loading*, o MonetDB foi de 2 a 12 vezes, aproximadamente, mais lento do que o PostgreSQL. No entanto, levando em conta somente os resultados do MonetDB, verifica-se que os tempos de execução obtidos por meio de *bulk loading* chegaram a ser mais de 126 vezes menores do que os tempos de execução obtidos por meio de operações comuns de inserção. Dessa forma, é possível obter tempos aceitáveis em operações de carga no MonetDB, já que a carga é feita uma única vez e o tempo gasto pode ser amortizado pelo ganho obtido durante a execução das consultas.

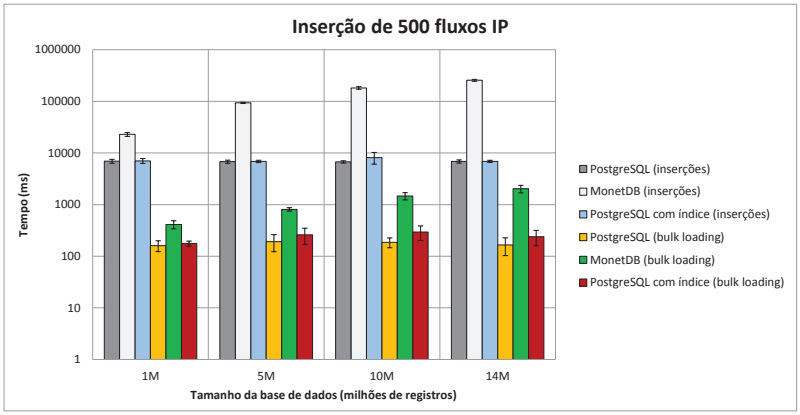


Figura 13. Inserção de 500 fluxos IP.

5. Conclusão

Este artigo apresentou a proposta de utilizar-se bancos de dados orientados a coluna para aumentar o desempenho de análises sobre grandes bases de fluxos IP. Na maioria das consultas realizadas nos experimentos, o MonetDB apresentou desempenho superior ao PostgreSQL. Em alguns casos, o PostgreSQL com índice foi mais rápido. Porém, quando os dados a serem manipulados são mais volumosos, nos casos em que a seletividade da consulta é menor, o MonetDB leva vantagem inclusive sobre o PostgreSQL com índice. Os resultados obtidos através deste trabalho mostram o potencial de soluções de armazenamento orientadas a coluna para a manipulação de dados de fluxos IP. Soluções como essa podem ser utilizadas no desenvolvimento de novas ferramentas a fim de otimizar o gerenciamento de dados em ambientes de gerência de redes.

6. Agradecimentos

Os autores agradecem à CAPES o apoio financeiro para o desenvolvimento deste trabalho.

Referências

Balman, M. and Byna, S. (2011). Open Problems in Network-Aware Data Management in Exa-Scale Computing and Terabit Networking Era. In *Proceedings of the 1st International Workshop on Network-Aware Data Management*, NDM '11, pages 73–78, New York, NY, USA. ACM.

Claise, B. (2004). Cisco Systems NetFlow Services Export Version 9. RFC 3954 (Informational).

Claise, B., Trammell, B., and Aitken, P. (2013). Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC 7011 (Standards Track).

Clemm, A. (2006). *Network Management Fundamentals*. Cisco Press.

Deri, L., Chou, E., Cherian, Z., Karmarkar, K., and Patterson, M. (2011). Increasing Data Center Network Visibility with Cisco NetFlow-Lite. In *Proceedings of the 7th International Conference on Network and Services Management*, CNSM '11, pages 274–279, Laxenburg, Austria, Austria. International Federation for Information Processing.

Harizopoulos, S., Liang, V., Abadi, D. J., and Madden, S. (2006). Performance Tradeoffs in Read-Optimized Databases. In *Proceedings of the 32nd International Conference on Very Large Data Bases*, VLDB '06, pages 487–498. VLDB Endowment.

Liu, W. and Zhou, Y. (2013). A Kind of Memory Database Engine Based on Column-Storage Techniques. In *Proceedings of the 8th International Conference on Computer Science Education*, ICCSE '13, pages 436–440.

Liu, Z., He, B., Hsiao, H.-I., and Chen, Y. (2011). Efficient and Scalable Data Evolution with Column-Oriented Databases. In *Proceedings of the 14th International Conference on Extending Database Technology*, EDBT/ICDT '11, pages 105–116, New York, NY, USA. ACM.

Min, H. and Franke, H. (2010). Improving In-Memory Column-Store Database Predicate Evaluation Performance on Multi-Core Systems. In *Proceedings of the 22nd International Symposium on Computer Architecture and High Performance Computing*, SBAC-PAD '10, pages 63–70, Washington, DC, USA. IEEE Computer Society.

Sperotto, A., Sadre, R., Vliet, F., and Pras, A. (2009). A Labeled Data Set for Flow-Based Intrusion Detection. In *Proceedings of the 9th IEEE International Workshop on IP Operations and Management*, IPOM '09, pages 39–50, Berlin, Heidelberg. Springer-Verlag.

Stonebraker, M., Abadi, D. J., Batkin, A., Chen, X., Cherniack, M., Ferreira, M., Lau, E., Lin, A., Madden, S., O'Neil, E., O'Neil, P., Rasin, A., Tran, N., and Zdonik, S. (2005). C-Store: A Column-Oriented DBMS. In *Proceedings of the 31st International Conference on Very Large Data Bases*, VLDB '05, pages 553–564. VLDB Endowment.

Zhu, H., Zhang, X., and Ding, W. (2011). Research on Errors of Utilized Bandwidth Measured by NetFlow. In *Proceedings of the 2nd International Conference on Networking and Distributed Computing*, ICNDC '11, pages 45–49.