Processamento eficiente de consultas sobre grandes volumes de dados usando arquiteturas multi-core

Frank W. R. da Silva[†,¥], Victor T. de Almeida[†,§], Vanessa Braganholo[†]
[†]Instituto de Computação, Universidade Federal Fluminense (UFF)
[¥]Universidade do Estado de Mato Grosso (UNEMAT)
[§]Petrobras S.A.
{frankwrs, valmeida, vanessa}@ic.uff.br
Programa de Pós-graduação em Ciência da Computação – IC/UFF
Nível: Mestrado
Ingresso: Março/2016
Previsão de Conclusão: Fevereiro/2018

Abstract. *Big Data Management Systems usually manage each machine as one node in the parallel query processing pipeline. In multi-core architectures, they leave several processor cores aside that could contribute to speed-up query processing. In this context, this dissertation contributes by exploring the use of all available processor cores, assessing the query processing performance in several scenarios. In particular, we use the concept of worker nodes (which are allocated in cores without disk access) and data nodes (which are allocated in cores with disk access) in the same machine using the MyriaX engine as a base platform that supports this concept. We evaluate several cluster configurations varying the amount of data and worker nodes to process queries and workloads with and without data replication factors. Preliminary results show that increasing the I/O parallelism in terms of data nodes is not always the most effective strategy, but adding worker nodes in available processing cores do improve speed-up up to certain levels. This reinforces the idea of using worker nodes in the query processing pipeline.*

1. Introdução

A necessidade de análise de grandes volumes de dados continua a crescer na indústria, governo e ciência. Isto levou empresas e organizações a buscar alternativas com melhor custo/benefício, pois sistemas tradicionais de banco de dados tornaram-se opções pouco atrativas. Motivadas pela emergência de plataformas de nuvem que dependem de *clusters* de centenas ou milhares de máquinas de propósito geral e também pelo aumento do poder de computação com a tecnologia *multi-core*, estas organizações contribuíram para o surgimento de diversos sistemas de gerência de Big Data.

Estes sistemas, em sua maioria, utilizam *clusters* de máquinas de propósito geral para processamento massivamente paralelo de consultas. As arquiteturas utilizadas por eles variam. Alguns exigem que cada máquina tenha um disco de dados acoplado, tais como ElasTras [Das et al., 2013], Impala [Bittorf et al., 2015], HadoopDB [Abouzeid et al., 2009], Apache Spark[1], Pregel [Malewicz et al., 2010], Dryad [Isard et al., 2007], Asterix [Alsubaiee et al., 2012], Presto[2], MyriaX [Wang et al., 2017] e Vertica[3]. Já o SnowFlake [Dageville et al., 2016] utiliza-se de uma base de dados compartilhada onde todos os nós acessam de forma concorrente. Outros sistemas exploram a tecnologia *multi-core* alocando vários nós em uma mesma máquina, tais como Nephele [Warneke and Kao, 2009], Amazon Redshift [Gupta et al., 2015] e Greenplum[4]. Contudo, em tais sistemas o recurso de armazenamento é fatiado ou compartilhado entre os nós, o que implica em concorrência no acesso ao disco.

A maioria destes sistemas trata cada máquina como sendo um único nó. Mesmo que cada máquina possua diversos processadores, cada um com diversos núcleos, e vários discos de dados, estes são normalmente gerenciados pelo sistema como um único recurso. No entanto, sabe-se que o uso de mais núcleos implica um melhor potencial de escalabilidade para operadores relacionais, tais como junções [Kim et al., 2009]. Contudo, em nossa pesquisa bibliográfica, não encontramos qualquer estratégia que explore totalmente os recursos disponíveis de armazenamento e processamento de forma independente em uma mesma máquina, sendo incapazes de explorar núcleos ociosos de processadores para atuar no processamento de operadores da consulta.

Com a finalidade de melhorar a eficiência do processamento paralelo de consultas em sistemas de gerência de Big Data, esta dissertação investiga o impacto da exploração de arquiteturas *multi-core* e replicação de dados no processamento paralelo de consultas. Em especial, investigamos: (i) o impacto da utilização de todos os recursos de processamento (núcleos dos processadores) disponíveis por máquina no tempo de processamento de consultas; (ii) o impacto do uso de replicação de dados, comparando o desempenho do cenário (i) com e sem replicação; e (iii) o impacto do uso das técnicas dos cenários (i) e (ii) em execução de conjuntos de consultas (*workloads*). Em nossas análises, a exploração de todos os recursos de processamento é feita por meio da alocação, em uma mesma máquina, de *worker nodes*, que realizam processamento, mas não possuem acesso a disco, e *data nodes*, que processam e acessam dados em disco. A partir da avaliação experimental desta abordagem, pretende-se definir um conjunto de

[1] https://spark.apache.org/
[2] http://prestodb.io/
[3] https://www.vertica.com/
[4] http://greenplum.org/

heurísticas a serem usadas para alocação de recursos no processamento paralelo de consultas de alto custo em sistemas de gerência de Big Data.

O restante deste artigo apresenta a abordagem proposta (Seção 2), descreve os experimentos (Seção 3), resultados e avaliações preliminares (Seção 4) e finalmente discute algumas considerações finais (Seção 5).

2. Abordagem Proposta

Para avaliar o impacto do uso de *worker nodes (wn)* e *data nodes (dn)*, explorando recursos ociosos de processamento, no desempenho de consultas, utilizamos o mecanismo de execução de consultas relacional, *shared-nothing* e paralelo MyriaX, componente do sistema de gerência de Big Data Myria [Wang et al., 2017]. Sua arquitetura é composta por um *master node*, por *worker nodes* e *data nodes*. Esta escolha foi motivada pelo fato deste mecanismo permitir o uso de *worker nodes* e *data nodes* de forma independente. MyriaX gerencia cada nó como sendo de um único tipo (*worker node*). A diferenciação entre *data node* e *worker node* acontece na atribuição de tarefas para cada nó. Nós que recebem operações de leitura e/ou escrita de dados atuam como *data nodes*. Por outro lado, nós que não recebem tarefas de leitura e/ou escrita de dados atuam como *worker nodes*. Desse modo, o MyriaX viabiliza a alocação de *data nodes* e *worker nodes* em uma mesma máquina, apesar disto nunca ter sido explorado pelo Myria. A Figura 1 ilustra uma implantação deste mecanismo com nove máquinas, sendo quatro *data nodes* e quatro *worker nodes*, além do *master node*.

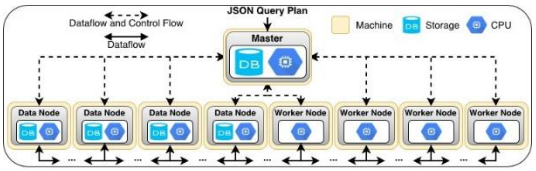

Figura 1. Arquitetura MyriaX com nove máquinas

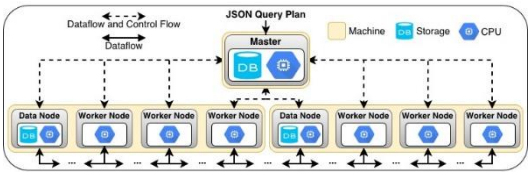

Figura 2. Arquitetura MyriaX com três máquinas

A Figura 1 ilustra o possível cenário *m:8_dn:4_wn:4* e a Figura 2 o cenário *m:2_dn:2_wn:6*. O cenário da Figura 1 utiliza 8 máquinas (*m:8*) para processamento, organizadas em 4 *data nodes* (*dn:4*) e 4 *worker nodes (wn:4)*, além do *master node*. Como os dados são particionados entre os data nodes usando uma estratégia de *round-robin* [Mehta and DeWitt, 1997], este cenário oferece um maior particionamento dos dados em relação ao cenário da Figura 2, que conta com 2 *data nodes* e 6 *worker nodes*, além do *master node*. Este último cenário, por sua vez, sofre menor influência de tráfego de rede na transmissão de dados entre nós, pois contém 2 máquinas com 1 *data node* e 3 *worker nodes* cada, aumentando assim o paralelismo intra-máquina. Ambos os cenários, após concluídas as operações de leitura de dados (realizada apenas por *data nodes*), terão 8 nós (2 *dn* + 6 *wn*) disponíveis para o processamento paralelo da consulta.

Em nossa avaliação experimental, planejamos a realização de três experimentos atacando diferentes pontos importantes ao desempenho do processamento paralelo de consultas analíticas. O primeiro (i) aplica diversas configurações e alocações de *data nodes* e *worker nodes* para processamento de consultas. O segundo experimento (ii) utiliza as configurações com melhor desempenho no primeiro e aplica diferentes fatores de replicação de dados entre *data nodes*. Por fim, o terceiro experimento (iii) utiliza *workloads* aplicadas às configurações com melhor desempenho do primeiro

experimento utilizando variações no fator de replicação de dados. Os detalhes do planejamento dos experimentos são apresentados na próxima seção.

3. Descrição dos Experimentos

Como mencionado na introdução, máquinas utilizadas no processamento paralelo de consultas podem conter recursos subutilizados, como núcleos de processadores com e sem discos acoplados, que podem ser utilizados como *worker nodes* e *data nodes* adicionais no processamento de consultas. Com base nisto, esta seção descreve o planejamento da avaliação experimental desta dissertação.

3.1. Experimento (i): explorar núcleos de processamento

Para validar a hipótese de que *"é possível diminuir o tempo de processamento de consultas através da adição de worker nodes em núcleos ociosos de processadores*", realizamos diversos experimentos preliminares em um *cluster*. O *cluster* utilizado é composto de 42 máquinas, cada uma com 2 processadores Intel Xeon *quadcore* (8 núcleos), 16 GB de RAM e 160 GB de HD. A rede é interligada com *switch Gigabit Ethernet*. O sistema operacional é o Red Hat Enterprise Linux Server versão 5.3.

Nesses experimentos, utilizamos um algoritmo que, a partir de uma quantidade de *worker nodes* e *data nodes*, gera vários cenários de distribuição e alocação de *data nodes* e *worker nodes* em máquinas do *cluster*. O script realiza 5 rodadas de consultas para cada cenário gerado. Cada rodada de consultas é executada para todos os cenários gerados antes de iniciar a rodada seguinte, ao invés de todas as 5 rodadas de consultas serem executadas de forma sequencial para cada cenário. Com esta lógica, evita-se que um cenário seja totalmente prejudicado por alguma atividade de manutenção do sistema operacional que fuja ao nosso controle durante a execução das consultas.

Cada cenário é composto por uma quantidade de máquinas (*m*), *data nodes* (*dn*) e *worker nodes* (*wn*), onde a quantidade de *worker nodes*, quando maior que zero, é dividida igualmente para cada máquina do cenário. A heurística para criar os cenários se baseou em valores de potência de 2 (iniciando em 2) para a quantidade de nós em cada implantação do serviço Myria em *cluster*, tendo como limite a quantidade máxima de núcleos de processadores disponibilizados, que para esse experimento foi um processador com 8 núcleos. Em cada implantação o algoritmo determina a quantidade para cada tipo de nó. Para *data nodes*, também foram utilizados valores de potência de 2 (iniciando em 2), tendo como limite a quantidade de máquinas, pois cada máquina possui uma única controladora de disco de dados. Isso evita a concorrência no acesso a disco. Os demais nós são alocados como *worker nodes*. Vale ressaltar que nos experimentos, *data nodes* também atuam como *worker nodes* no processamento da consulta. Em todos os cenários, uma das máquinas é reservada para atuar como *master node* e, portanto, apenas as demais 8 máquinas são consideradas pelos cenários. Cabe ressaltar que os dados são particionados entre os *data nodes* usando a estratégia de *round-robin* [Mehta and DeWitt, 1997], de forma que todos os *data nodes* acessam partições de mesmo tamanho.

A base de dados utilizada para o experimento preliminar foi a do Twitter que contém uma tabela com duas colunas (*follower* e *followee*) com valores de identificadores de usuários e representa relações entre usuários seguidores (*follower*) e seguidos (*followee*). Esta base contém 4.532.185 de tuplas e foi escolhida por ser uma base de dados real com grande volume de dados.

Foram utilizadas duas consultas referentes à base de dados do Twitter. A primeira consulta (C1) realiza auto-junção entre as colunas citadas da base, com a finalidade de identificar relações entre usuários onde ambos seguem um ao outro. A segunda consulta (C2) conta com uma operação de junção a mais e identifica triângulos entre usuários. Um triângulo se forma quando um usuário *A* segue um usuário *B*, que por sua vez segue um usuário *C*, que segue o usuário *A*. De fato, esta é uma consulta que ainda apresenta desafios para a comunidade científica quando os dados não cabem na memória [Hu et al., 2013]. Na base de dados utilizada no experimento, a primeira e segunda junções retornam 2.045.216.395 e 89.084.893 de tuplas, respectivamente. Durante as submissões das consultas, o algoritmo captura o tempo que cada consulta leva para executar, elimina o maior e menor tempo, e calcula a média dos tempos restantes. Isso é feito para cada cenário.

Para dar sequência aos experimentos discutidos nesta seção, planejamos utilizar *workloads* para avaliar o comportamento da abordagem proposta com execução de consultas que utilizam um número maior de tabelas e relações.

3.2. Experimento (ii): aplicar fator de replicação de dados

Planos de execução de consultas analíticas, em sua maioria, envolvem múltiplas operações de leitura de dados. Em nossa abordagem, essas operações são executadas por *data nodes*. Neste sentido, para validar a hipótese de que "*é possível diminuir o tempo de processamento de consultas que realizam mais de uma leitura de dados na mesma tabela, separando esta leitura em data nodes distintos, através da replicação de dados, aumentando o paralelismo de I/O*", planejamos reproduzir a execução das consultas utilizadas no experimento (i) utilizando diversos fatores de replicação de dados. Para tal, planejamos utilizar cenários com melhores desempenhos identificados no experimento (i). Os resultados serão avaliados e comparados com os resultados do primeiro experimento, onde não usamos replicação de dados.

3.3. Experimento (iii): utilizar *workloads*

Sistemas de gerência de Big Data atuam diretamente com *workloads*. Tais *workloads* envolvem múltiplas consultas diferentes executando em paralelo, algumas na mesma máquina, sobre uma massiva quantidade de dados. Neste sentido, o objetivo deste experimento é validar a hipótese de que "*é possível obter melhor eficiência no processamento de workloads utilizando configurações de worker nodes e data nodes e replicação de dados adequadas"*. Para tal, planejamos utilizar cenários de destaque do experimento (i) com fatores de replicação de dados identificados no experimento (ii) para execuções em *cluster*. Será realizado um estudo prévio para determinar qual *workload* será utilizada neste experimento.

4. Resultados e Avaliações Preliminares

Nesta seção são apresentados e avaliados os resultados preliminares obtidos com o experimento (i), pois os experimentos (ii) e (iii) estão em fase de planejamento. Por questões de espaço, os resultados obtidos com a consulta C2 não são apresentados neste artigo (apresentamos apenas uma breve avaliação dos resultados). Os resultados e avaliação completa são apresentados em [Silva et al., 2017]. Para fins de identificação, cenários padrão do MyriaX estão nomeados como *Baseline* e cenários que se baseiam na hipótese desse trabalho são nomeados *Avaliação*.

Consulta C1 (auto-junção). A Figura 3 apresenta os resultados para a consulta C1, utilizando cenários com até 8 máquinas agrupados pela quantidade total de nós (8, 16, 32 e 64). Os tempos são medidos em segundos.

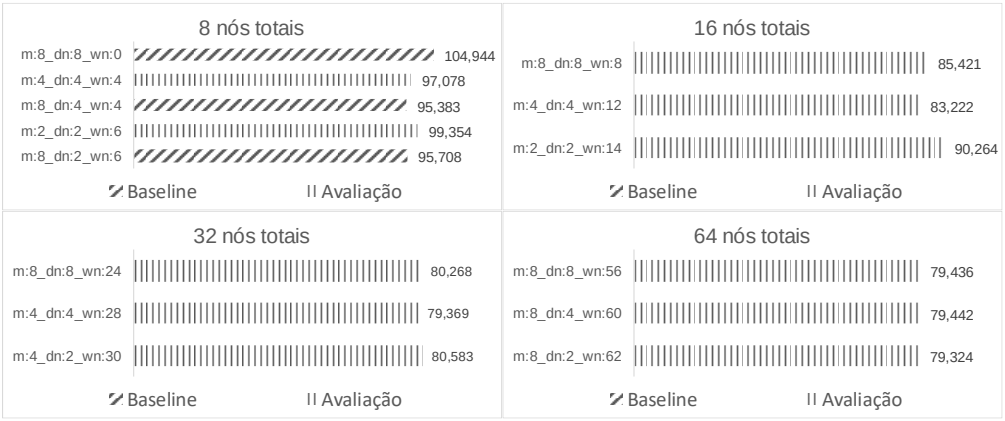

Figura 3. Tempo médio (em segundos) da consulta C1 em cenários com até 8 máquinas agrupados pela quantidade total de nós

Como apresenta a Figura 3, cenários com 8 nós totais apresentaram o maior tempo médio para execução da consulta C1. O cenário que não utiliza *worker nodes* apresentou o maior tempo médio dentre estes cenários, justificado pela execução sequencial dos operadores da consulta. Os demais cenários com quantidade totais de 16, 32 e 64 nós apresentam melhora gradativa no tempo médio do processamento da consulta C1, respectivamente, e apresentam resultados semelhantes dentro destas quantidades de nós totais. Isto porque estes cenários utilizam de maior quantidade de *worker nodes* que passam a processar pequenas quantidades de dados e aumentam o tráfego de rede. Desta forma, a Figura 3 evidencia que, para a consulta C1, é possível atingir um determinado desempenho com menor quantidade de máquinas dimensionando *workers nodes* e *data nodes* e explorando recursos de núcleos de processamento e discos de dados de uma mesma máquina. Por exemplo, o cenário *m:8_dn:4_wn:4* utiliza 8 máquinas e apresentou aceleração de apenas 1,05x e 1,2x em relação aos cenários *m:2_dn:2_wn:14* e *m:4_dn:4_wn:28* que utilizam 2 e 4 máquinas, respectivamente. É importante acrescentar que em experimentos adicionais, para os cenários com até 2 máquinas *m:2_dn:2_wn:0* e *m:2_dn:2_wn:14*, a simples adição de *worker nodes* implicou em aceleração de até 2,92x quando se explora todos os núcleos de processamento disponíveis.

Consulta C2 (triângulos). Para esta consulta, o uso de todo o recurso disponível em dada quantidade máquinas piora o desempenho de processamento. Por outro lado, cenários *Avaliação* com quantidades iguais de *data nodes* e *worker nodes* apresentaram melhores desempenhos, com aceleração de até 1,07x.

5. Considerações Finais

Esta dissertação avalia o uso de *worker nodes*, que participam do *pipeline* de processamento da consulta e não acessam dados em disco, e *data nodes*, que acessam dados em disco e também atuam como *worker nodes*, alocados em uma mesma máquina. O mecanismo de execução de consulta relacional, *shared-nothing* e paralelo MyriaX foi utilizado como plataforma base dos experimentos preliminares por permitir o uso desta abordagem. Em especial, nesse trabalho pretendemos investigar: (i) o impacto da utilização de todos os recursos de processamento disponíveis por máquina

no tempo de processamento de consultas através da alocação de *worker nodes*; (ii) o impacto do uso de replicação de dados, comparando o desempenho do cenário (i) com e sem replicação; e (iii) o impacto do uso das técnicas dos cenários (i) e (ii) em execução de *workloads*. Cada experimento da fase de avaliação experimental será descrito e avaliado em um ou mais artigos a serem submetidos, contribuindo para comunidade de pesquisa em bancos de dados paralelos. O experimento (i), como mostram as Seções 3 e 4, foi executado e avaliado. Os resultados deste foram descritos em um artigo completo [Silva et al., 2017] aceito para publicação no Simpósio Brasileiro de Banco de Dados (SBBD) 2017. A partir da avaliação experimental, pretende-se definir um conjunto de heurísticas a serem usadas para alocação de recursos no processamento paralelo de consultas de alto custo em sistemas de gerência de Big Data.

Referências

Abouzeid, A., Bajda-Pawlikowski, K., Abadi, D., Silberschatz, A. and Rasin, A. (2009). HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. *Proceedings of the VLDB Endowment (PVLDB)*, v. 2, n. 1, p. 922–933.

Alsubaiee, S., Behm, A., Grover, R., et al. (2012). ASTERIX: Scalable Warehouse-Style Web Data Integration. *International Workshop on Information Integration on the Web*, p. 1–4.

Bittorf, M., Bobrovytsky, T., Erickson, C. C. A. C. J., et al. (2015). Impala: A Modern, Open-Source SQL Engine for Hadoop. In *Conference on Innovative Data Systems Research (CIDR)*.

Dageville, B., Cruanes, T., Zukowski, M., et al. (2016). The Snowflake Elastic Data Warehouse. *International Conference on Management of Data (SIGMOD)*, p. 215–226.

Das, S., Agrawal, D. and El Abbadi, A. (2013). ElasTraS: An Elastic, Scalable, and Self-Managing Transactional Database for the Cloud. *Transactions on Database Systems (TODS)*, v. 38, n. 1, p. 5.

Gupta, A., Agarwal, D., Tan, D., et al. (2015). Amazon Redshift and the Case for Simpler Data Warehouses. In *International Conference on Management of Data (SIGMOD)*.

Hu, X., Tao, Y. and Chung, C.-W. (2013). Massive Graph Triangulation. In *SIGMOD*.

Isard, M., Budiu, M., Yu, Y., Birrell, A. and Fetterly, D. (2007). Dryad: Distributed Data-parallel Programs from Sequential Building Blocks. In *European Conference on Computer Systems (EuroSys)*.

Kim, C., Kaldewey, T., Lee, V. W., et al. (2009). Sort vs. Hash Revisited: Fast Join Implementation on Modern Multi-core CPUs. *Proceedings of the VLDB Endowment (PVLDB)*, v. 2, n. 2, p. 1378–1389.

Malewicz, G., Austern, M. H., Bik, A. J., et al. (2010). Pregel: A System for Large-scale Graph Processing. In *International Conference on Management of Data (SIGMOD)*.

Mehta, M. and DeWitt, D. J. (1997). Data Placement in Shared-nothing Parallel Database Systems. *The VLDB Journal*, v. 6, n. 1, p. 53–72.

Silva, F. W. R., Almeida, V. T. and Braganholo, V. (2017). Explorando Arquiteturas Multi-core para Processamento Eficiente de Consultas em Sistemas de Gerência de Big Data. In *Simpósio Brasileiro de Banco de Dados (SBBD)*.

Wang, J., Baker, T., Balazinska, M., et al. (2017). The Myria Big Data Management and Analytics System and Cloud Service. In *Conference on Innovative Data Systems Research (CIDR)*.

Warneke, D. and Kao, O. (2009). Nephele: Efficient Parallel Data Processing in the Cloud. In *Many-Task Computing on Grids and Supercomputers (MTAGS)*.