Estratégia para fase de Merge utilizando agrupamento local

Darlon Vasata[1,2], **Liria Matsumoto Sato**[1]

[1]Departamento de Engenharia Elétrica – Escola Politécnica da Universidade de São Paulo
Av. Professor Luciano Gualberto s/n - CEP 05508-110
São Paulo, SP, Brasil

[2]Instituto Federal do Paraná – Câmpus Cascavel
Av. das Pombas, 2020 – CEP 85814-800
Cascavel, PR, Brasil

darlon.vasata@ifpr.edu.br, liria.sato@poli.usp.br

Abstract. *The processing of great quantities of data is a subject extensively discussed by the scientific community. Models like PMDE and Mapreduce process those data in phases, which involve execution, data exchanging and grouping. The Merge phase is responsible for grouping values associated with the same key. In this work, we propose to divide this phase in two steps: tree size reduction grouping pairs with local merge and merge of pairs received from other machines. The strategy will be integrated to the PMDE execution environment. We intend to obtain performance for specific application using this strategy, decreasing data traffic and reducing memory use.*

Resumo. *O processamento de aplicações de grandes quantidades de dados é um tema amplamente discutido pela comunidade científica atualmente. Modelos como PMDE e Mapreduce processam tais dados em fases, que envolvem execução, trocas de dados e agrupamento. A fase de Merge é responsável por agrupar valores associados a uma mesma chave. Neste trabalho, propomos dividir tal fase em duas etapas: redução do tamanho da árvore utilizando agrupamento de pares com merge local e merge de pares recebidos de outras máquinas. A estratégia será aplicada ao ambiente de execução do modelo PMDE. Pretende-se obter um aumento de desempenho para determinadas aplicações utilizando esta estratégia, por meio da redução no tráfego de dados e diminuição do uso de memória.*

Introdução

Um dos grandes temas discutidos atualmente na comunidade científica é a temática em torno do fenômeno da geração e processamento de grandes quantidades de dados, que popularizou-se como Big Data[McAfee and Brynjolfsson 2012]. As vantagens do processamento de grandes quantidades de dados são diversas e abrangem áreas como análise de comportamento de usuários, estratégias de marketing, análises de DNA, dentre outros.

Um dos desafios envolvendo Big Data é a forma de processamento dos dados. Visto que se tratam de quantidades de dados na ordem de Gigabytes ou Terabytes, o processamento de tais dados com as tecnologias atuais de processamento e memória pode demandar muito tempo. Para tal finalidade, diferentes modelos de programação utilizando sistemas distribuídos foram desenvolvidos tais como Mapreduce

[Dean and Ghemawat 2008] e PMDE [Vasata and Sato 2017]. Nestes modelos, o processamento é realizado em fases. Na fase de Map, o usuário define um código para processamento dos dados e geração de pares do tipo (k, v), onde k é uma chave e v um valor associado a k. A fase de Merge consiste em agrupar todos os valores associados a uma mesma chave e repassá-los à fase de Reduce, no formato $(k, \{v_0, v_1, ..., v_n\})$. Nesta, os dados são processados e a saída é obtida. Uma das características destes modelos é sua escalabilidade e facilidade de programação, visto que a troca de dados entre máquinas e agrupamento de valores são realizados de maneira automática. O usuário deve apenas fornecer o código a ser executado nas fases de Map e Reduce, bem como definir configurações de execução no ambiente. Neste trabalho, propõe-se dividir a fase de Merge em etapas, com objetivo de proporcionar aumento de desempenho na execução das aplicações que utilizam grandes quantidades de dados. Com o uso da estratégia, espera-se que menos dados sejam trafegados e uma menor quantidade de memória seja utilizada pelo ambiente de execução. Pretende-se aplicar a estratégia apresentada neste trabalho no ambiente de execução PMDE, observar os resultados obtidos e compará-los com o desempenho atual da ferramenta.

Estratégia de Merge utilizando agrupamento local

A fase de Merge é necessária nos modelos de execução para processamento de grandes quantidades de dados. A maneira como é realizada pode impactar no desempenho das aplicações. O PMDE é um modelo de execução e ambiente de programação distribuído para grandes quantidades de dados, que utiliza blocos [Vasata and Sato 2015] de execução, entrada de dados e saída de dados com múltiplas entradas e saídas seguindo um DAG (Grafo Acíclico Direto) para o processamento das aplicações. No ambiente PMDE, a fase de Merge é implementada em duas etapas. Neste trabalho pretendemos aplicar na primeira etapa um procedimento semelhante ao da segunda etapa, com o intuito de reduzir a quantidade de memória utilizada e o número de dados trafegados.

Um dos problemas relacionados à fase de Merge é o tempo necessário para o seu processamento. Comumente nesses sistemas os pares (k, v) são organizados utilizando estruturas de árvore e a manipulação de elementos nessas estruturas implica em alto tempo de processamento, dado pelo grande número de pares existentes. Para que o tamanho da árvore seja diminuído, é possível agrupar alguns pares, como ocorre na primeira etapa. Porém organizar estes pares leva a outro problema, que é percorrê-los e ordená-los, para que posteriormente possam ser agrupados.

A estratégia apresentada neste trabalho consiste em diminuir o tamanho da árvore e agrupar pares com a mesma chave quando um determinado limiar for alcançado, além de dividir a fase de Merge em duas etapas, apresentadas e discutidas nas subseções a seguir.

Primeira etapa: redução do tamanho da árvore com agrupamento local

A primeira etapa consiste na utilização de conjuntos de pares com objetivo de reduzir o tamanho da árvore, em que eles são armazenados nos nós da árvore. Os pares são organizados no conjunto utilizando estruturas de lista, sendo que o critério para que pertençam ao conjunto é definido com base nos b bits mais à esquerda do *hash* da chave k. A Figura 1 apresenta um exemplo em que os pares pertencem a um mesmo conjunto de acordo com

os primeiros 8 bits do *hash*, bem como é demonstrado o problema da colisão de *hashes*, a ser tratado nas próximas etapas.

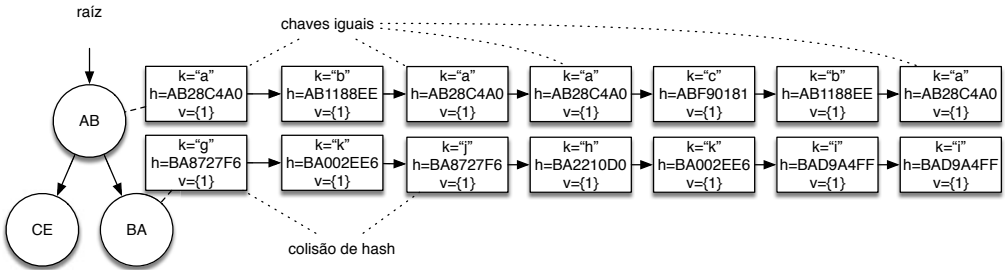

Figura 1. Primeira etapa - redução do tamanho da árvore

O foco central deste trabalho consiste em agrupar os valores v associados à mesma chave k em um mesmo conjunto, ainda durante a primeira etapa, que pode ocorrer mais de uma vez, de acordo com algum limiar alcançado, como número de pares no conjunto ou pela quantidade de memória utilizada, seja em um único ou em todos os conjuntos da árvore, ou então de maneira contínua por uma *thread* auxiliar. O objetivo principal deste processo é diminuir a memória utilizada por chaves repetidas no mesmo conjunto. Os valores são agrupados sem a utilização de um algoritmo de ordenação, empregando um vetor de endereços para estruturas de pares (k, v), de forma que a posição do vetor é calculada com base no *hash* da chave aplicado ao *mod* do tamanho do vetor. A lista de pares é percorrida, e caso a posição do vetor já esteja ocupada, verifica-se se possuem a mesma chave. Caso sim, os valores são agrupados, e caso não, uma colisão ocorreu e o par sendo processado é adicionado a uma lista auxiliar a ser processada posteriormente. Levando em conta que todo o grupo é transferido a outras máquinas utilizando uma mesma mensagem, a redução do número de pares no grupo implica também em uma menor quantidade de dados a serem transferidos. Não é pretendido que essa etapa seja obrigatória, principalmente em situações em que o número de pares no conjunto não seja elevado. A Figura 2 demonstra um exemplo da aplicação desta etapa, para os pares presentes no conjunto que possuem *hash* iniciando pelo valor `AB`, apresentados na Figura 1. Os *hashes* das chaves foram omitidos na imagem.

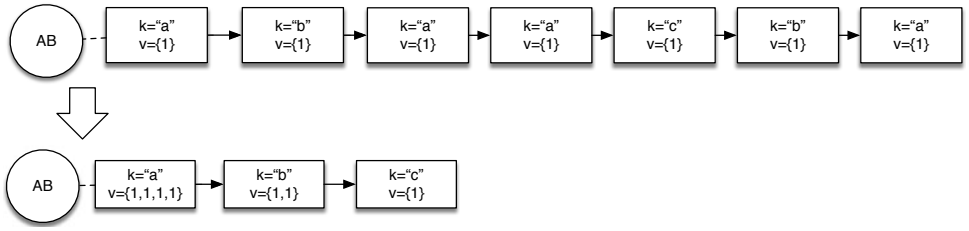

Figura 2. Primeira etapa - Merge local

Segunda etapa - Merge de todos os pares recebidos

Após a troca de dados, o procedimento de agrupar valores com a mesma chave é aplicado novamente, visto que são recebidos conjuntos de pares de outras máquinas. A Figura 3 ilustra esse agrupamento, destacando pares recebidos de outras máquinas.

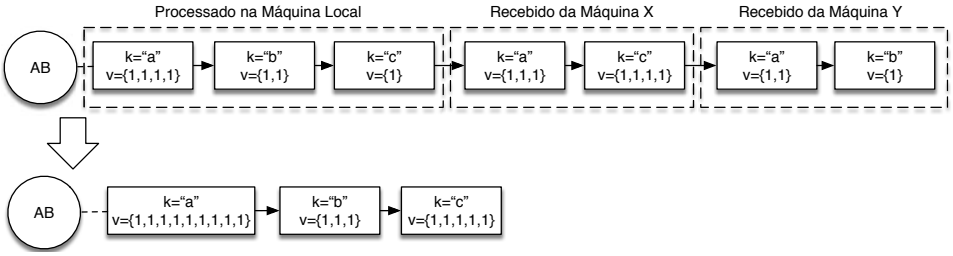

Figura 3. Segunda etapa - Merge de todos os pares recebidos

Avaliação

Como avaliação dos resultados, pretende-se analisar o tempo necessário para o processamento em cada etapa, a quantidade de memória utilizada e o tamanho ideal do vetor de endereços utilizado para o agrupamento de pares.

Ainda está em processo de análise qual deve ser o melhor instante para aplicar o agrupamento de pares local, se este deve ser feito de acordo com o número de pares ou conforme a quantidade de memória utilizada, de forma a obter melhores desempenhos para as aplicações.

Resultados esperados

Espera-se que com a aplicação da estratégia de Merge em etapas o desempenho das aplicações implementadas utilizando o modelo PMDE seja superior. Acredita-se que a aplicação da estratégia terá efeito positivo no desempenho em aplicações que possuem alto número de chaves iguais sendo processadas na mesma máquina. Quanto aos instantes em que a segunda etapa deve ser executada, espera-se que um melhor desempenho seja atingido aplicando o agrupamento apenas aos conjuntos com um alto número de pares.

Conclusão

Neste trabalho foi apresentada uma estratégia para a fase de Merge utilizando duas etapas, a ser implementada junto à ferramenta PMDE. Espera-se que com esta estratégia as aplicações implementadas seguindo este modelo possuam aumento de desempenho.

Agradecimentos

Os autores agradecem ao Instituto Federal do Paraná e à Escola Politécnica da Universidade de São Paulo pelo suporte às pesquisas realizadas.

Referências

Dean, J. and Ghemawat, S. (2008). MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM*, 51(1):107–113.

McAfee, A. and Brynjolfsson, E. (2012). Big data: the management revolution. *Harvard business review*, 90(10):60–6, 68, 128.

Vasata, D. and Sato, L. M. (2015). Mapreduce com múltiplos blocos de execução atuando de forma coordenada utilizando entradas e saídas interligadas. In *Anais do VI ERAD-SP*, São José do Rio Preto.

Vasata, D. and Sato, L. M. (2017). Modelo de Execução Paralelo em Ambiente Distribuído. Não publicado.