Centro Federal de Educação Tecnológica de Minas Gerais
Departamento de Computação
Engenharia de Computação

Mariane Raquel Silva Gonçalves

Comparação de algoritmos paralelos de ordenação em MapReduce

Orientadora: Prof[a]. Dr[a]. Cristina Duarte Murta

Belo Horizonte
2012

Sumário

1 Introdução

1.1 Definição do Problema

As tendências atuais em design de microprocessadores estão mudando fundamentalmente a maneira que o desempenho é obtido a partir de sistemas computacionais. A indústria declarou que seu futuro está em computação paralela, com o aumento crescente do número de núcleos dos processadores [Asanovic et al., 2009]. O modelo de programação *single core* (sequencial) está sendo substituído rapidamente pelo modelo *multi-core* (paralelo), e com isso surge a necessidade de escrever software para sistemas com multiprocessadores e memória compartilhada [Ernst et al., 2009].

Arquiteturas *multi-core* podem oferecer um aumento significativo de desempenho sobre as arquiteturas de núcleo único, de acordo com as tarefas paralelizadas. No entanto, muitas vezes isto exige novos paradigmas de programação para utilizar eficientemente a arquitetura envolvida [Prinslow, 2011]. A ordenação é um exemplo de tarefa que pode ter seu desempenho melhorado com o uso de paralelismo. A ordenação paralela é o processo de reorganizar uma sequência de entrada e produzir uma saída ordenada de acordo com um atributo através de múltiplas unidades de processamento, que ordenam em conjunto a sequência de entrada.

Uso crescente de computação paralela em sistemas computacionais gera a necessidade de algoritmos de ordenação inovadores, desenvolvidos para dar suporte a essas aplicações.

1.2 Motivação

O limite físico de aumento na frequência de operação dos processadores levou a indústria de hardware a substituir o processador de núcleo único por vários processadores eficientes em um mesmo *chip*, os processadores *multi-core*. É preciso, então, criar aplicações que utilizem efetivamente o crescente número de núcleos dos

processadores [Asanovic et al., 2009].

Um grande número de aplicações paralelas possui uma fase de computação intensa, na qual uma lista de elementos deve ser ordenada com base em algum de seus atributos. Um exemplo é o algoritmo de Page Rank [Page et al., 1999] da Google: as páginas de resultado de uma consulta são classificadas de acordo com sua relevância, e então precisam ser ordenadas de maneira eficiente [Kale and Solomonik, 2010].

Na criação de algoritmos de ordenação paralela, é ponto fundamental ordenar coletivamente os dados de cada processo individual, de forma a utilizar todas as unidades de processamento e minimizar os custos de redistribuição de chaves entre os processadores. Fatores como movimentação de dados, balanço de carga, latência de comunicação e distribuição inicial das chaves são considerados ingredientes chave para o bom desempenho da ordenação paralela, e variam de acordo com o algoritmo escolhido como solução[Kale and Solomonik, 2010]. No exemplo do Page Rank, o número de páginas a serem ordenadas é enorme, e elas são recolhidas de diversos servidores da Google; é uma questão fundamental escolher algoritmo paralelo com o melhor desempenho dentre as soluções possíveis.

O MapReduce[Dean and Ghemawat, 2008] é um modelo de programação paralela criado pela Google para processamento de grandes volumes de dados em *clusters*. Esse modelo propõe simplificar a computação paralela e ser de fácil uso, abstraindo conceitos complexos da paralelização - como tolerância a falhas, distribuição de dados e balanço de carga - e utilizando duas funções principais: map e reduce. A complexidade do algoritmo paralelo não é vista pelo desenvolvedor, que pode se ocupar em desenvolver a solução proposta. O Hadoop [White, 2009] é uma das implementações do MapReduce, um *framework open source* desenvolvido por Doug Cutting em 2005 que provê o gerenciamento de computação distribuída.

O desenvolvimento de soluções capazes de lidar com grandes volumes de dados é uma das preocupações atuais, tendo em vista a quantidade de dados processados diariamente, e o rápido crescimento desse volume de dados. Não é fácil medir o volume total de dados armazenados digitamente, mas uma estimativa da IDC colocou o tamanho do "universo digital"em 0,18 zettabytes em 2006, e previa um crescimento dez vezes até 2011 (para 1,8 zettabytes). *The New York Stock Exchange* gera cerca de um terabyte de novos dados comerciais por dia. O Facebook armazena aproximadamente 10 bilhões de fotos, que ocupam mais de um petabyte.

The Internet Archive armazena aproximadamente 2 petabytes de dados, com aumento de 20 terabytes por mês [White, 2009]. Estima-se que dados não estruturados são a maior porção e de a mais rápido crescimento dentro das empresas, o que torna o processamento de tal volume de dados muitas vezes inviável.

MapReduce e sua implementação *open source* Hadoop representam uma alternativa economicamente atraente oferecendo uma plataforma eficiente de computação distribuída para lidar com grandes volumes de dados e mineração de petabytes de informações não estruturadas [Cherkasova, 2011].

Devido ao grande número de algoritmos de ordenação paralela e variadas arquiteturas paralelas, estudos experimentais assumem uma importância crescente na avaliação e seleção de algoritmos apropriados para multiprocessadores.

1.3 Objetivos

Os objetivos deste trabalho são:

- Estudar a programação paralela aplicada à algoritmos de ordenação;
- Implementar um ou mais algoritmos de ordenação paralela no modelo MapReduce, com o software Hadoop;
- Comparar duas ou mais implementações de algoritmos paralelos de ordenação.

O trabalho desenvolvido por [de Morais Pinhão, 2011] apresentou um estudo sobre a computação paralela e algoritmos de ordenação no modelo MapReduce, através da implementação do algoritmo de Ordenação por Amostragem feita em ambiente Hadoop.

Este projeto busca continuar o estudo sobre ordenação paralela feito no trabalho citado, com a da análise de desempenho de dois ou mais algoritmos de ordenação - sendo um deles o algoritmo de ordenação por amostragem. A análise busca compará-los com relação à quantidade de dados a serem ordenados, variabilidade dos dados de entrada e número máquinas utilizadas.

1.4 Estado da Arte

1.5 Organização

2 Referencial Teórico

2.1 Computação Paralela

2.2 Ordenação Paralela

Condições de implementação de algoritmos paralelos de ordenação

Habilidade de explorar distribuições iniciais parcialmente ordenadas Alguns algoritmos podem se beneficiar de cenários nos quais a sequência de entrada dos dados é mesma, ou pouco alterada. Nesse caso, é possível obter melhor desempenho ao realizar menos trabalho e movimentação de dados. Se a alteração na posição dos elementos da sequência é pequena o suficiente, grande parte dos processadores mantém seus dados iniciais e precisa se comunicar apenas com os processadores vizinhos.

Movimentação dos dados A movimentação de dados entre processadores deve ser mínima durante a execução do algoritmo. Em um sistema de memória distribuída, a quantidade de dados a ser movimentada é um ponto crítico, pois o custo de troca de dados pode dominar o custo de execução total e limitar a escalabilidade.

Balanceamento de carga O algoritmo de ordenação paralela deve assegurar o balanceamento de carga ao distribuir os dados entre os processadores. Cada processador deve receber uma parcela equilibrada dos dados para ordenar, uma vez que o tempo de execução da aplicação é tipicamente limitada pela execução do processador mais sobrecarregado.

Latência de comunicação A latência de comunicação é definida como o tempo médio necessário para enviar uma mensagem de um processador a outro. Em grandes sistemas distribuídos, reduzir o tempo de latência se torna muito importante.

Sobreposição de comunicação e computação Em qualquer aplicação paralela, existem tarefas com focos em computação e comunicação. A sobreposição de tais tarefas permite que sejam feitas tarefas de processamento e ao mesmo tempo operações de entrada e saída de dados, evitando que os recursos fiquem ociosos durante o intervalo de tempo necessário para a transmissão da carga de trabalho.

2.3 Algoritmos

2.3.1 Sample Sort

2.3.2 Quick Sort

3 Metodologia

3.1 Recursos necessários

3.2 Desenvolvimento //

3.3 Descrição dos experimentos

4 Resultados Preliminares

5 Conclusões e Propostas de Continuidade

Referências Bibliográficas

Asanovic, K., Bodik, R., Demmel, J., Keaveny, T., Keutzer, K., Kubiatowicz, J., Morgan, N., Patterson, D., Sen, K., Wawrzynek, J., Wessel, D., and Yelick, K. 2009. A view of the parallel computing landscape. *Commun. ACM 52,* 10 (Oct.), 56–67.

Cherkasova, L. 2011. Performance modeling in mapreduce environments: challenges and opportunities. In *ICPE'11.* 5–6.

de Morais Pinhão, P. 2011. Ordenação paralela no ambiente hadoop.

Dean, J. and Ghemawat, S. 2008. Mapreduce: simplified data processing on large clusters. *Commun. ACM 51,* 1 (Jan.), 107–113.

Ernst, D., Wittman, B., Harvey, B., Murphy, T., and Wrinn, M. 2009. Preparing students for ubiquitous parallelism. In *SIGCSE.* 136–137.

Kale, V. and Solomonik, E. 2010. Parallel sorting pattern. In *Proceedings of the 2010 Workshop on Parallel Programming Patterns.* ParaPLoP '10. ACM, New York, NY, USA, 10:1–10:12.

Page, L., Brin, S., Motwani, R., and Winograd, T. 1999. The pagerank citation ranking: Bringing order to the web.

Prinslow, G. 2011. Overview of performance measurement and analytical modeling techniques for multi-core processors.

White, T. 2009. *Hadoop: The Definitive Guide*, first edition ed. O'Reilly.