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

A ordenação é um dos problemas fundamentais da ciência da computação e e um dos problemas algorítmicos mais estudados. Muitas aplicações dependem de ordenações eficientes como base para a seu próprio desempenho. A ordenação por si só é um problema que abrange desde sistemas de banco de dados à computação gráfica, e muitos outros algoritmos podem ser descritos em termos de ordenação [11, Amato 1996].

Na última década, a quantidade de dados (de trabalho, utilizada pelos sistemas, disponíveis) aumentou várias ordens de grandeza, fazendo o processamento dos dados um desafio para a computação sequencial. Como resultado, torna-se crucial substituir a computação tradicional por computação distribuída eficiente [Lin e Dyer 2010]. A mudança no modelo de programação sequencial para paralelo é um fato inevitável e ocorre gradualmente, desde que a indústria declarou que seu futuro está em computação paralela, com o aumento crescente do número de núcleos dos processadores [Asanovic 2009].

Uso crescente de computação paralela em sistemas computacionais gera a necessidade de algoritmos de ordenação inovadores, desenvolvidos para dar suporte a essas aplicações. Consequentemente, é importante desenvolver rotinas eficientes de ordenação em arquiteturas paralelas e distribuídas.

O MapReduce é um modelo de programação paralela criado pela Google para processamento de grandes volumes de dados em *clusters* [Dean e Ghemawat 2008]. 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. O Hadoop é uma implementação do modelo MapReduce e um *framework* para desenvolvimento de aplicações paralelas. O trabalho proposto por **(author?)**.

- crescimento dos dados
- map reduce como proposta para processamento rápido e fácil de grandes quantidades de dados
- uso do modelo map reduce em ordenação
- comparação de algoritmos de ordenação nesse modelo

1.2 Motivação

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.

O aumento no poder computacional, nas últimas décadas, se deve, largamente, (Inovação e avanço nas comunidades científicas e enter-prêmio ter sido alimentado pela melhora, implacável exponencial da capacidade de hardware do computador nos últimos 40 anos. A força motriz para grande parte dessa melhoria tem sido a capacidade de dobrar o número de dispositivos microeletrônicos em uma área constante de silício a um custo quase constante aproximadamente a cada dois anos. Esta melhoria exponencial na contagem do transistor a cada dois anos é amplamente conhecido como Lei de Moore.)

A dissipação de energia em dispositivos digitais com clock é proporcional à freqüência de clock, impondo um limite natural em taxas de clock. podemos esperar uma melhora muito pequena no desempenho de série de CPUs de propósito geral. O aumento no desempenho virá em vez de computação paralela.

[Manferdelli 2008]

Arquitetos de computador têm sido forçados a recorrer a arquiteturas paralelas para continuar a fazer progressos. [Manferdelli 2008]

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 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 2009].

É um caminho natural para o processamento de dados em larga escala o uso de clusters.

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].

O MapReduce[Dean e 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.

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].

A classificação é um dos problemas fundamentais da ciência da computa-

ção e algoritmos paralelos para classificação têm sido estudados desde o início da computação paralela.

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 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 e Solomonik 2010].

Os algoritmos ótimos existentes em arquitetura sequencial, como Quick sort e Heap Sort necessitam de um tempo mínimo *(n log n)* para ordenar uma sequência de n elementos. [AHO 1700]

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 e 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.

Dado o grande número de algoritmos de ordenação paralelas e uma vasta variedade de arquiteturas paralelas, é uma tarefa difícil escolher o melhor algoritmo para uma determinada máquina e instância do problema. A principal razão que a escolha é mais difícil do que em máquinas sequenciais é porque não existe um modelo teórico conhecido que pode ser aplicado para prever com precisão o desempenho de um algoritmo em arquitecturas diferentes. Assim, exper-imental estudos assumem uma crescente importância para a avaliação e seleção de algoritmos apropriados para multiprocessadores. Tem havido um número de estudos de implementação relatados na literatura nos últimos anos (ver, por exemplo, [5, 12]). No entanto, mais estudos são necessários antes que possamos aproximar-se do ponto onde um determinado algoritmo pode ser recomendado para uma determinada máquina com algum grau de confiança. [Amato 1996]

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 **(author?)** 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 Organização do Texto

Esse projeto está organizado em 6 capítulos. O próximo capítulo apresenta o referencial teórico para o desenvolvimento do trabalho. O capítulo 3 complementa o referencial teórico, e apresenta de maneira mais detalhada os conceitos mais importantes da ordenação em ambiente paralelo. O capítulo 4 descreve a metologia de pesquisa à ser aplicada no desenvolvimento do projeto. Os resultados preliminares obtidos até a entrega do projeto são apresentados no capítulo 5. As conclusões obtidas até o momento e os próximos passos para a conclusão do projeto estão no 6.

2 Referencial Teórico

- computação paralela
- modelos de computação paralela (memória compartilhada, distribuida, threads, paralelismo de dados e map reduce)
- map reduce
- ordenação

3 Ordenação Paralela

- importância da ordenação paralela
- formas de ordenação: memória e disco
- grandes dados: apenas disco
- grandes dados: problematização (tempo, limite de memória)
- grandes dados: sort benchmark

- algoritmos de ordenação paralelos
- funcionamento geral
- condições / ingredientes / limites
- diferentes algoritmos para diferentes aplicações
- descrição de algoritmos (e diagramas): sample sort, quick sort

3.1 Algoritmos de 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.

3.1.1 Sample Sort

3.1.2 Quick Sort

4 Metodologia

4.1 Recursos necessários

4.2 Desenvolvimento //

4.3 Descrição dos experimentos

5 Resultados Preliminares

6 Conclusões e Propostas de Continuidade

Referências Bibliográficas

[Amato 1996]Amato, Nancy M. et al. *Y.: A comparison of parallel sorting algorithms on different architectures*. [S.l.], 1996.

[Asanovic 2009]Asanovic, Krste et al. A view of the parallel computing landscape. *Commun. ACM*, ACM, New York, NY, USA, v. 52, n. 10, p. 56–67, out. 2009. ISSN 0001-0782.

[Cherkasova 2011]Cherkasova, Ludmila. Performance modeling in mapreduce environments: challenges and opportunities. In: *ICPE'11*. [S.l.: s.n.], 2011. p. 5–6.

[Dean e Ghemawat 2008]Dean, Jeffrey; Ghemawat, Sanjay. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, ACM, New York, NY, USA, v. 51, n. 1, p. 107–113, jan. 2008. ISSN 0001-0782.

[Ernst 2009]Ernst, Daniel et al. Preparing students for ubiquitous parallelism. In: *SIGCSE*. [S.l.: s.n.], 2009. p. 136–137.

[Kale e Solomonik 2010]Kale, Vivek; Solomonik, Edgar. Parallel sorting pattern. In: *Proceedings of the 2010 Workshop on Parallel Programming Patterns*. New York, NY, USA: ACM, 2010. (ParaPLoP '10), p. 10:1–10:12. ISBN 978-1-4503-0127-5.

[Lin e Dyer 2010]Lin, Jimmy; Dyer, Chris. *Data-Intensive Text Processing with MapReduce*. [S.l.]: Morgan & Claypool Publishers, 2010. (Synthesis Lectures on Human Language Technologies).

[Morais Pinhão 2011]de Morais Pinhão, Paula. *Ordenação Paralela no Ambiente Hadoop*. 2011.

[Page 1999]Page, Lawrence et al. *The PageRank Citation Ranking: Bringing Order to the Web*. 1999.

[Prinslow 2011]Prinslow, Garrison. *Overview of Performance Measurement and Analytical Modeling Techniques for Multi-core Processors*. 2011.

[11]Satish, Nadathur; Harris, Mark; Garland, Michael. Designing efficient sorting algorithms for manycore gpus. In: *Proceedings of the 2009 IEEE International Symposium on Parallel&Distributed Processing*. Washington, DC, USA: IEEE Computer Society, 2009. (IPDPS '09), p. 1–10. ISBN 978-1-4244-3751-1. Disponível em: <http://dx.doi.org/10.1109/IPDPS.2009.5161005>.

[White 2009]White, Tom. *Hadoop: The Definitive Guide*. first edition. O'Reilly, 2009. Disponível em: <http://oreilly.com/catalog/9780596521981>.