Análise de Desempenho da Ferramenta Apache Mahout para Mineração de Dados Distribuída

Adriano Pereira[1], **Andrea Schwertner Charão**[1], **César A. F. De Rose**[2]

[1]Laboratório de Sistemas de Computação (LSC)
Universidade Federal de Santa Maria (UFSM)

[2]Laboratório de Alto Desempenho (LAD)
Pontifícia Universidade Católica do RS (PUCRS)

{apereira,andrea}@inf.ufsm.br, cesar.derose@pucrs.br

Resumo. *Este trabalho apresenta uma investigação do desempenho da ferramenta Apache Mahout, para a mineração de dados distribuída e escalável, utilizando o modelo de programação MapReduce. Foram analisados os algoritmos K-Means e FPGrowth, em um ambiente distribuído, onde se alterou o número de nodos de processamento, para verificar o desempenho e a escalabilidade da ferramenta.*

1. Introdução

O avanço das tecnologias computacionais impulsionou a indústria de informação, tornando disponíveis grandes volumes de dados. Com isso, surge o interesse em analisar teorias e ferramentas capazes de extrair novas informações e conhecimentos úteis a partir dessas fontes de dados [Fayyad et al. 1996]. Neste ponto está a mineração de dados, processo de análise de grandes volumes de dados, em busca de relações implícitas, a partir das quais pode-se extrair informações novas, compreensíveis e úteis [Hand et al. 2001].

Técnicas centralizadas de mineração de dados tornam-se inadequadas frente à complexidade das tarefas e à distribuíção física das informações. Como alternativa, pode-se utilizar múltiplos processadores e bases de dados, em um ambiente distribuído, para acelerar o processo de mineração [Pérez et al. 2007]. A ferramenta Apache Mahout implementa um conjunto de algoritmos de mineração de dados distribuída, fazendo uso do modelo MapReduce, que promete escalabilidade ao dividir a carga de trabalho em tarefas independentes, podendo ser executadas paralelamente. Para isso, Mahout faz uso da infraestrutura provida pela ferramenta Apache Hadoop [Apache Mahout 2010]. Este trabalho faz uma análise do desempenho do Apache Mahout para a mineração de dados distribuída, verificando o comportamento de escalabilidade da ferramenta conforme aumenta-se o conjunto de nodos disponíveis para o processamento. O restante deste artigo expõe a ferramenta Apache Mahout, os casos de teste realizados e a avaliação do desempenho obtido.

2. Apache Mahout

Apache Mahout é uma biblioteca de algoritmos de mineração de dados distribuída, escrita em linguagem de programação Java [Apache Mahout 2010]. Seus algoritmos são escritos com o modelo de programação MapReduce, fazendo uso da infraestrutura da ferramenta Apache Hadoop, um conjunto de programas para a computação distribuída e escalável

[Apache Hadoop 2007]. Dentre os programas do Hadoop, Mahout utiliza o sistema de arquivos distribuído *Hadoop Distributed File System* (HDFS) e o *framework* MapReduce.

MapReduce é um modelo de programação inspirado nas primitivas *map* e *reduce* das linguagens de programação funcionais. Seu funcionamento consiste em dividir os dados de entrada em conjuntos independentes e aplicar, em cada conjunto, uma operação *map*, gerando um conjunto de pares intermediários chave/valor. Após, aplica-se uma uma operação *reduce*, agrupando valores cujos pares possuem a mesma chave. Os algoritmos do Mahout utilizam o *framework* do Hadoop que implementa este modelo de programação. Além disso, o uso do HDFS, pelo Apache Mahout, possibilita a distribuição dos dados analisados pelos algoritmos de mineração.

Apache Mahout suporta quatro classes de operações: (i) recomendação, em que se recomendam itens, a partir do comportamento dos usuários; (ii) agrupamento, onde registros são divididos em classes (*clusters*), com base em sua semelhança; (iii) classificação, em que se classificam itens com base em uma categorização existente; (iv) mineração por frequência de grupos de itens, onde se identificam padrões e relações em grupos de itens [Apache Mahout 2010].

3. Desenvolvimento

Primeiramente escolheram-se dois algoritmos de mineração implementados pelo Mahout. Em seguida, configurou-se um ambiente distribuído e prepararam-se dois conjuntos de dados, para analisar o desempenho da ferramenta. Os algoritmos foram executados no ambiente configurado, analisando os dados preparados, verificando o ganho de desempenho ao adicionar-se nodos ao processamento.

Os algoritmos escolhidos para a análise do desempenho da ferramenta foram: *K-Means* e *Parallel FPGrowth* (FPG). **K-Means** é um algoritmo de agrupamento, escolhido por ser classificado como simples e efetivo [Larose 2005]. **FPG** é um algoritmo de mineração de padrões frequentes. Foi escolhido por ser considerado eficiente e escalável, para mineração de grandes e pequenos padrões [Borgelt 2005].

Para ser executado em um ambiente distribuído, Apache Mahout necessita que este ambiente esteja configurado com o Apache Hadoop. Usualmente, define-se um nodo mestre no sistema distribuído, o qual executará instâncias de *NameNode*, o mestre do HDFS, que gerencia a árvore de arquivos e diretórios; e *JobTracker*, mestre do MapReduce, que divide os *jobs* – unidades de trabalho – em *tasks* do tipo map e reduce, e as escalona nos escravos. Cada escravo possui uma instância de *DataNode*, o qual gerencia os blocos de arquivos do HDFS, e *TaskTracker*, que executa as *tasks* de map e reduce.

Os dados utilizados neste trabalho são provenientes de um sistema de informações acadêmicas, do Centro de Processamento de Dados, da Universidade Federal de Santa Maria, estando em formato CSV. É necessário transformá-los, para que sejam corretamente utilizados como entrada dos algoritmos de mineração, implementados pelo Mahout. Os algoritmos de agrupamento, como o K-Means, utilizam um formato de vetor n-dimensional para os dados de entrada. A ferramenta possui formas automatizadas para a criação dos vetores; dentre elas, tem-se a conversão de arquivos em formato ARFF, utilizado pela ferramenta Weka [ARFF 2008], para o formato vetorial, a qual foi utilizada neste trabalho. Desenvolveu-se um aplicativo para converter os arquivos originais para o

formato ARFF, que foram convertidos, posteriormente, para o formato vetorial, através de uma das funcionalidades do Mahout. Para o algoritmo FPGrowth, o formato esperado é semelhante a um CSV, mas sem cabeçalho, e com um caractere especial utilizado para separar os campos. Desenvolveu-se um aplicativo para realizar essa conversão.

4. Casos de Teste e Resultados

Para verificar o desempenho do Mahout, utilizou-se um ambiente de teste com plataforma GNU/Linux. Trata-se de um *cluster* do Laboratório de Alto Desempenho da Pontifícia Universidade Católica do Rio Grande do Sul, nomeado **Gates**. O *cluster* Gates é composto por 6 máquinas, com processador AMD Opteron Processor 246 de 2.00GHz de *clock*, 1MB de memória *cache* e 8GB memória de RAM. Utilizou-se um núcleo por nodo. Alterou-se o número de nodos, em cada caso de teste, partindo de uma execução centralizada, com 1 nodo, até execuções distribuídas, utilizando de 2 a 6 nodos. Nas execuções distribuídas, definiu-se uma máquina como mestre, e as demais como escravos. Definiram-se dois casos de teste, que são descritos a seguir.

O primeiro caso consistiu na execução do algoritmo K-Means, para o agrupamento em 5 classes (*clusters*), com um máximo de 5 iterações, utilizando como entrada um arquivo de 157.6MB, composto por 961635 vetores de 45 dimensões. Os resultados obtidos estão resumidos na tabela 1. Verificou-se que houve ganho de desempenho com a inclusão de nós ao processamento. Os ganhos foram mais significativos até a execução distribuída com 4 nodos, havendo uma diminuição na taxa de crescimento, com o acrescimo de nodos, visualizado pela curva de aceleração da figura 1. Neste caso, observou-se um comportamento de escalabilidade na ferramenta.

Tabela 1. Desempenho K-Means

Número de Nodos	Tempo de Execução(ms)	Aceleração
1 (Local)	399163	1
2	374279	1,0664
3	299217	1,3340
4	276054	1,4459
5	269658	1,4802
6	261492	1,5264

O segundo caso de teste consistiu na execução do algoritmo FPGrowth, eliminando valores com frequência menor que 2, e encontrando os 50 padrões mais altos. Utilizaram-se dados compostos por 9984 registros de 52 campos, totalizando 6MB de arquivo texto. A tabela 2 sumariza os resultados obtidos. Neste caso, não houve ganho de desempenho ao adicionar-se nodos ao processamento, o que pode ser visto na curva ilustrada na figura 1. Verificando-se os rastros de execução, percebeu-se que apenas uma task foi criada e escalonada a cada instante. Dessa forma, a execução foi sequencial, e não tirou proveito do paralelismo oferecido pelo modelo MapReduce.

5. Conclusão

Neste trabalho foi apresentado um estudo acerca do desempenho da ferramenta Apache Mahout, para mineração de dados distribuída, verificando, especialmente, a relação de ganho ao se adicionar nodos ao processamento distribuído. Verificaram-se dois algoritmos,

Tabela 2. Desempenho FPGrowth

Nº de Nodos/Tasks	Tempo de Execução (ms)	Aceleração
1 (Local)	2391794	1
2	2553442	0,9366
3	2551015	0,9375
4	2515010	0,9510
5	2574364	0,9290
6	2565894	0,9321

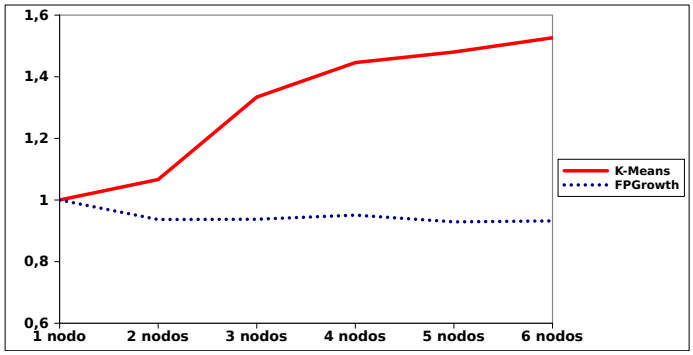

Figura 1. Curvas de Aceleração dos Algoritmos

K-Means e FPGrowth, sendo executados em um *cluster*, onde se alterou o número de nodos disponíveis para a execução. Visualizaram-se diferenças quanto ao comportanto dos algoritmos analisados. O K-Means obteve ganho de desempenho ao adicionar-se nodos ao processamento, visualizando-se um comportamento de escalabilidade. A execução do algoritmo FPGrowth não obteve melhoras com o aumento do número de nodos de processamento; verificou-se que este algoritmo possui uma execução sequencial, embora faça uso do modelo de programação MapReduce.

Referências

Apache Hadoop (2007). Welcome to Apache Hadoop! Disponível em: http://hadoop.apache.org Acesso em: setembro de 2010.

Apache Mahout (2010). Apache Mahout: Scalable machine-learning and data-mining library. Disponível em: http://mahout.apache.org Acesso em: setembro de 2010.

ARFF (2008). Attribute-relation file format (ARFF). Disponível em: http://www.cs.waikato.ac.nz/ ml/weka/arff.html. Acesso em: novembro de 2010.

Borgelt, C. (2005). An implementation of the FP-growth algorithm. In *OSDM '05: Proceedings of the 1st international workshop on open source data mining*. ACM.

Fayyad, U. M., Piatetsky-Shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery: an overview. *Advances in knowledge discovery and data mining*.

Hand, D., Mannila, H., and Smyth, P. (2001). *Principles of Data Mining*.

Larose, D. T. (2005). *Discovering Knowledge in Data: An Introduction to Data Mining*.

Pérez, M. S., Sánchez, A., Robles, V., Herrero, P., and Peña, J. M. (2007). Design and implementation of a data mining grid-aware architecture. *Future Gener. Comput. Syst.*