Experimentos de Mineração de Dados Paralela e Distribuída com Grid Weka

Vinícius Garcia Pinto[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)

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

Resumo. *Este trabalho apresenta experimentos com a ferramenta para mineração de dados paralela e distribuída Grid Weka, baseada no ambiente centralizado Weka. São descritos dois casos de teste, com os algoritmos J48 e KStar, visando analisar o desempenho da ferramenta Grid Weka quando executada de forma local e distribuída, variando-se a configuração do ambiente computacional.*

1. Introdução

A mineração de dados consiste na aplicação de algoritmos sobre grandes bases de dados com a finalidade de extrair conhecimento útil não trivial dessas [Fayyad et al. 1996]. O tamanho e a heterogeneidade das bases de dados associados à complexidade dos algoritmos de mineração de dados tornam necessária a utilização de soluções paralelas e distribuídas para acelerar e viabilizar o processo de mineração de dados [Fu 2001, Pérez et al. 2007].

A ferramenta Grid Weka [Zuo et al. 2004] é uma implementação modificada do ambiente Weka para permitir a execução de algumas tarefas do processo de mineração de dados de forma paralela e distribuída. O Weka [Witten et al. 1999] é um ambiente que agrega diferentes algoritmos e técnicas para mineração de dados de forma centralizada, sendo utilizado como base para ferramentas que buscam prover distribuição e paralelismo. Este trabalho apresenta um estudo de caso da ferramenta Grid Weka identificando técnicas implementadas, a disponibilidade de paralelismo e distribuição bem como uma análise do desempenho da ferramenta quando variada a configuração do ambiente computacional utilizado. Não foram encontrados trabalhos semelhantes que apresentem análise do Grid Weka ou dados atualizados sobre ferramentas similares [Bernardi 2010]. As demais seções deste artigo apresentam as características da mineração de dados com Grid Weka, os casos de teste realizados e a conclusão do trabalho.

2. Mineração de Dados com Grid Weka

A ferramenta Grid Weka faz uso do código dos algoritmos de mineração de dados disponibilizados pelo Weka para permitir a execução distribuída, explorando os recursos de vários computadores. O Grid Weka diferencia-se de outras ferramentas baseadas no Weka, como o Weka4WS [Talia et al. 2005], por não necessitar de middleware específico no ambiente computacional e por disponibilizar um número maior de tarefas para execução distribuída, quando comparado a ferramenta Weka-Parallel [Celis and Musicant 2002]. A execução é feita em uma grade *ad-hoc*, composta por nós servidores e por um ou mais

nó(s) cliente(s). Em máquinas multiprocessadas são colocadas em execução, paralelamente, várias instâncias servidoras do Grid Weka.

Identificou-se que a ferramenta permite a execução distribuída de uma parte das tarefas disponibilizadas pelo Weka, sendo elas: (i) execução remota do treinamento de um modelo de classificação e (ii) execução remota e paralela da validação cruzada e do teste de um modelo de classificação. Todas essas tarefas são relacionadas à técnica de classificação. Juntamente com a tarefa de treinamento é feita validação cruzada. Esta é realizada paralelamente entre o nó cliente e o(s) nó(s) servidor(es).

A entrada de dados para o Grid Weka é feita através de arquivos texto do tipo ARFF *(Attribute-Relation File Format)*, compostos por um cabeçalho com a declaração dos atributos e por uma seção de dados com as instâncias contendo os valores para os atributos. Os conjuntos de dados utilizados nesse trabalho são reais, provenientes do sistema de informações acadêmicas da UFSM e foram fornecidos pelo Centro de Processamento de Dados da instituição. Esses dados foram disponibilizados em arquivos texto do tipo CSV.

A conversão dos arquivos no formato CSV disponibilizado para o formato ARFF utilizado pela ferramenta foi feita utilizando a API do Weka 3.7 através das classes *CSVLoader*, *Instances* e *ArffSaver*. Para a realização da conversão foi necessária uma reformatação dos arquivos CSV para adequá-los ao formato esperado pela API utilizada. Os campos em branco foram marcados com "?", os caracteres não permitidos foram substituídos e as linhas quebradas foram reagrupadas. Os arquivos em formato ARFF obtidos foram particionados arquivos menores contendo frações do arquivo original, a fim de que os arquivos contendo um número menor de registros fossem utilizados na etapa de treinamento do modelo de classificação e os arquivos maiores fossem utilizados nas etapas de teste do modelo treinado.

3. Casos de Teste

Foram considerados dois casos de teste para a análise do desempenho da ferramenta Grid Weka. Utilizou-se um ambiente de testes dedicado com plataforma GNU/Linux composto por duas máquinas: uma delas equipada com dois processadores Intel Xeon E335 de 2Ghz com quatro núcleos cada e 4Gb de memória RAM; a outra máquina é equipada com um processador Intel Pentium IV com 3Ghz e 512Mb de memória RAM. Na máquina com processador Pentium IV foi colocada em execução a instância cliente do Grid Weka, enquanto na máquina com processadores Xeon foram colocadas em execução várias instâncias servidoras. Nestes casos de teste cada uma dessas instâncias será considerada como um servidor.

Os casos de teste compreendem duas etapas da técnica de classificação: a etapa de treinamento de um modelo de classificação e a etapa de teste do modelo de classificação treinado. No primeiro caso de teste (I) foi utilizado o algoritmo de classificação *J48*, um algoritmo do tipo árvore de decisão. No segundo caso de teste (II) foi utilizado o algoritmo de classificação *KStar*, um algoritmo baseado em exemplos que utiliza uma função de distância baseada na medida da entropia.

O conjunto de dados utilizado nos dois casos de teste é composto por 34 atributos, sendo que utilizou-se um destes atributos como campo para classificação. Este

conjunto possui 28519 instâncias. Utilizou-se um subconjunto com 408 instâncias, equivalente a 1/70 do conjunto total, para a execução da etapa de treinamento do modelo de classificação. Na etapa de teste do modelo de classificação utilizou-se o conjunto total.

As etapas de treinamento do modelo de classificação foram executadas de forma local e distribuída utilizando um, dois, três ou quatro servidores remotos. Dessa forma, um dos servidores realiza o treinamento, e o cliente e os demais servidores realizam a validação cruzada. A tabela 1 apresenta os tempos de execução obtidos para os casos de teste I e II.

Tabela 1. Etapas de treinamento do modelo de classificação

	Tempo Caso de Teste I (s)	Tempo Caso de Teste II (s)
Local	17,399s	23,692s
Remoto (1)	22,547s	31,426s
Remoto (2)	28,266s	34,415s
Remoto (3)	32,398s	37,831s
Remoto (4)	33,667s	39,315s

As etapas de teste do modelo de classificação foram executadas de forma local e distribuída utilizando um, dois, quatro ou seis servidores remotos. A tabela 2 apresenta os tempos de execução obtidos para esta etapa em ambos os casos de teste.

Tabela 2. Etapas de teste do modelo de classificação

	Tempo Caso de Teste I (s)	Tempo Caso de Teste II (s)
Local	23,253s	294,217s
Remoto (1)	74,773s	191,213s
Remoto (2)	95,082s	154,265s
Remoto (4)	112,519s	138,718s
Remoto (6)	165,613s	123,210s

4. Conclusão

Neste trabalho foi apresentado um estudo de caso da ferramenta para mineração de dados paralela e distribuída Grid Weka. Foram executados dois casos de teste com os algoritmos de classificação *J48* e *KStar* variando a forma de execução, local ou distribuída, e o número de servidores utilizados na execução distribuída. Em cada caso foi realizada uma etapa de treinamento e uma etapa de teste do modelo de classificação.

Através dos resultados de desempenho obtidos, observou-se que durante a etapa de treinamento do modelo de classificação, nos dois casos de teste, a execução remota utilizando a ferramenta Grid Weka não apresentou vantagens sobre a execução centralizada com relação ao tempo transcorrido para a execução dessa tarefa. Durante a etapa de teste do modelo de classificação no caso de teste I (utilizando o algoritmo *J48*) observou-se que também não houve melhora no tempo de processamento quando são adicionados servidores remotos. O tempo de execução da tarefa aumentou conforme aumentava o número de servidores remotos em utilização. Entretanto, no caso de teste II (utilizando o algoritmo *KStar*) o tempo de execução da tarefa de teste do modelo diminuiu conforme

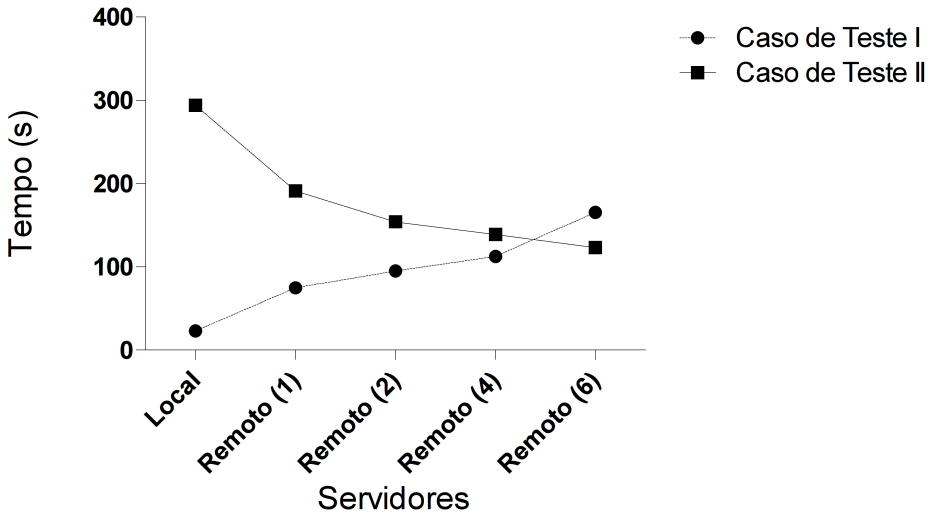

Figura 1. Etapas de teste do modelo de classificação

o aumento do número de servidores utilizados paralelamente, o que neste caso, torna a execução distribuída utilizando o Grid Weka mais rápida que a execução local.

Referências

Bernardi, E. F. F. (2010). Uma arquitetura para suporte à mineração de dados paralela e distribuída em ambientes de computação de alto desempenho.

Celis, S. and Musicant, D. R. (2002). Weka-parallel: Machine learning in parallel. Technical report, Department of Mathematics and Computer Science. Carleton College, Northfield.

Fayyad, U., Piatetsky-shapiro, G., and Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI Magazine*, 17:37–54.

Fu, Y. (2001). Distributed data mining: An overview. *IEEE Technical Committee on Distributed Processing newsletter*.

Pérez, M. S., Sánchez, A., Robles, V., Herrero, P., and Pe Na, J. M. (2007). Design and implementation of a data mining grid-aware architecture. *Future Generations Computer Systems*, 23(1):42–47.

Talia, D., Trunfio, P., and Verta, O. (2005). Weka4ws: a wsrf-enabled weka toolkit for distributed data mining on grids. In *Proc. of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2005*, pages 309–320. Springer-Verlag.

Witten, I. H., Frank, E., Trigg, L., Hall, M., Holmes, G., and Cunningham, S. J. (1999). Weka: Practical machine learning tools and techniques with java implementations.

Zuo, X., Khoussainov, R., and Kushmerick, N. (2004). Grid-enabled weka: A toolkit for machine learning on the grid. *ERCIM News*, 59.