Aplicação de um sistema para mineração de dados de vendas

Fábio R. Bot da Silva[1] silva.fabio@unicenp.edu.br
Gustavo A. Giménez-Lugo[1 2] gustavo_gl@unicenp.edu.br
Silvio Bortoleto[1] silvio.bortoleto@unicenp.edu.br

[1] UNICENP – Centro Universitário Positivo – Curitiba, PR, Brasil
[2] UTFPR – Universidade Tecnológica Federal do Paraná – Curitiba, PR, Brasil

RESUMO

Grandes volumes de dados requerem alto desempenho para viabilizar a descoberta de conhecimento. O presente trabalho descreve uma ferramenta de mineração distribuída, inspirada no algoritmo a priori (Agrawal e Srikant, 1994), na forma de um algoritmo de alto desempenho. Como estudo de caso é detalhado o processo de descoberta de combinações simultâneas em transações de vendas, determinando o grau de interesse dos conjuntos descobertos através da freqüência com que ocorrem.

Palavras-Chave: Mineração. Descoberta. Conhecimento. BI.

1. INTRODUÇÃO

Atualmente as empresas enfrentam um cenário de competição acirrada na disputa por nichos de mercado cada vez menores. A capacidade de adaptação ao mercado, buscando constantemente melhorar a eficiência, passou a ser elemento de sobrevivência das empresas. Segundo Kimball (2002) o poder das soluções de *Business Intelligence* pode proporcionar à organização uma substancial vantagem competitiva em relação a seus competidores.

Este artigo demonstra os resultados obtidos com a implementação de uma ferramenta distribuída para mineração de dados, cujo desenvolvimento foi motivado pela necessidade de obter informações estratégicas, visando o incremento de vendas. A ferramenta procura descobrir padrões de consumo através da análise dos conjuntos de vendas realizadas na mesma transação, determinando o grau de interesse do conjunto descoberto pela freqüencia com que o mesmo ocorre. Trata-se então, da descoberta de conjuntos freqüentes através da mineração dos dados de vendas.

A mineração destas informações em grandes bases de dados demanda uma capacidade de processamento expressiva, sendo que a memória do sistema tende a esgotar-se rapidamente, pois o volume de combinações a experimentar é muito grande, tornando a realização desta tarefa bastante árdua do ponto de vista computacional. A arquitetura utilizada é baseada no conceito do algoritmo *Apriori* (Agrawal e Srikant, 1994), mas com a diferença de separar o processamento dos conjuntos freqüentes conforme o número de elementos combinados, perdendo desta forma algumas características do algoritmo *APriori*. Por outro lado, esta separação permite distribuir o processamento, aumentando a capacidade da mineração e reduzindo o gargalo causado pela memória limitada dos equipamentos.

A seguir, após referenciar algumas técnicas de processamento distribuído, abordaremos o algoritmo *Apriori,* seguido do detalhamento desta ferramenta e do modelo de distribuição utilizado, concluindo com a apresentação e análise dos resultados.

2. TÉCNICAS DE PROCESSAMENTO PARALELO

O processamento paralelo consiste em aumentar o número de tarefas realizadas em um período de tempo, buscando melhorar o desempenho ou até fornecer a capacidade computacional necessária para a conclusão de processamentos pesados. Para isso, segundo

Morrison (2003), é preciso decompor o problema em partes menores, separando as tarefas ou cálculos que podem ser executados de forma independente para serem executados de forma paralela, em processadores ou núcleos de processamento distintos.

De uma forma simplificada, o processamento paralelo pode ser realizado com a utilização de equipamentos com mais de um processador (SMP - *Symetric Multi Processor*) ou mais de um núcleo de processamento na mesma máquina, neste caso, a memória e o barramento do sistema são compartilhados entre as unidades de processamento. Pode também ser feito por um aglomerado de computadores ligados em rede, que se comunicam através de um sistema de computação distribuída operando como se fosse uma única máquina de grande porte, esta modalidade é denominada *cluster*. É importante distinguir um sistema de multiprocessamento paralelo (SMP) de um sistema distribuído, no primeiro caso todas as unidades de processamento encontram-se na mesma máquina enquanto que no sistema distribuído (*cluster*) os processadores ou conjuntos de processadores estão separados fisicamente.

A seguir serão apresentadas algumas modalidades de *cluster* mais utilizadas:

Cluster Beowulf: Utiliza um conjunto de computadores (normalmente sem teclado, mouse e monitor) em modo exclusivo para o processamento distribuído, sendo que um dos equipamentos é destinado ao gerenciador do *cluster*. A conexão dos nós pode ser feita por redes padrão *Ethernet* e pode utilizar computadores comuns, até mesmo obsoletos. O sistema é baseado em Linux, no qual é instalado um conjunto de pacotes de software para controlar o *cluster*, é importante observar que não se trata de um sistema operacional de distribuição, mas de um conjunto de aplicações que rodam sobre o sistema operacional. Para que as aplicações possam tirar proveito desta modalidade de *cluster*, devem ser modificadas e recompiladas usando uma biblioteca para troca de mensagens como a PVM (*Parallel Virtual Machine*) ou MPI (*Message Passing Interface*). A correta modificação da aplicação introduzindo chamadas às funções destas bibliotecas é que permitem ao gerenciador do *cluster* realizar a correta distribuição das partes da aplicação, tendo impacto direto no desempenho do *cluster*. Esta modalidade distribui partes de um processo pai para serem processadas em paralelo nos computadores do *cluster*, é muito útil na execução de algoritmos que possuam cálculos sofisticados ou rotinas que exijam grande capacidade de processamento.

As vantagens desta modalidade são o baixo custo de implantação e o aproveitamento de equipamentos antigos e como desvantagens o fato de o desempenho ficar limitado ao processamento definido pelo equipamento gerenciador do *cluster*, além da modificação e recompilação das aplicações a serem distribuídas.

Cluster Mosix: Esta modalidade de *cluster* difere da anterior por distribuir processos e não partes de um processo pai. O sistema é composto por uma versão do Linux modificada de forma a suportar o processamento distribuído sendo que o software de distribuição é compilado juntamente ao Linux e não sendo um pacote de software que roda sobre o sistema operacional como uma aplicação. Esta modalidade é eficiente para tarefas de balanceamento de carga, como exigido por servidores Web e de e-mail. As aplicações devem ser modificadas para fazer comunicação entre processos, normalmente através de *pipes,* que é muito mais simples do que a correta aplicação das bibliotecas de mensagens no caso do Beowulf. Por exemplo, aplicações como o servidor Web Apache, que cria processos filhos conforme a necessidade passa a ter grande escalabilidade com o uso deste tipo de cluster, pois os processos criados vão sendo repassados aos nós do *cluster*, este é um caso típico de balanceamento de carga, onde uma pilha de processos é distribuída pela rede.

As vantagens, segundo Bueno (2002), são a facilidade na adaptação das aplicações para aproveitar o cluster sendo que muitas nem precisam ser alteradas e o melhor

aproveitamento do *cluster* devido à migração automática de processos entre os computadores da rede. As desvantagens são que como o sistema de distribuição consiste em um adendo ao *kernel*, é necessário recompilar o Linux para obter o sistema operacional do *cluster*.

Cluster de Workstation: Esta modalidade de *cluster* utiliza um conjunto de computadores completos (com teclado, mouse e monitor) conectados em rede e que servem tanto ao propósito de fornecer recursos computacionais para o processamento paralelo quanto para o uso diário em aplicações corriqueiras, como navegação na Internet, leitura de e-mails ou edição de textos e planilhas. As aplicações a serem distribuídas requerem o uso de bibliotecas de troca de mensagens para permitir o processamento paralelo, sendo que a execução de uma aplicação cliente habilita a estação de trabalho a operar como um nó do *cluster*.

A vantagem é o melhor aproveitamento dos computadores, já que o mesmo computador pode atender a dois propósitos, a desvantagem é que a divisão da capacidade de processamento do computador, uma parte para o uso normal e outra para auxiliar ao cluster deixará ambas as atividades mais lentas, a liberação da capacidade integral do equipamento para o cluster ficará reduzido aos finais de expediente e fins de semana.

Quanto ao desenvolvimento do software para suportar o processamento paralelo, pode-se dizer que conforme a situação existe uma forma que melhor se adapta, por exemplo, se dispomos de um equipamento com mais de um processador (SMP), a aplicação deverá ser escrita de modo a criar processos filhos, implementando recursos para comunicação entre processos ou então criar várias linhas de execução concorrentes (threads) de forma a aproveitar o recurso de multiprocessamento. Já a modalidade de *cluster* Mosix beneficia-se somente do conceito de múltiplos processos, a página inicial do projeto menciona "f*ork and forget*", ou seja, crie processos filhos e esqueça, deixando claro que Mosix sabe muito bem o que fazer com múltiplos processos. O Beowulf e o *cluster* de Workstation exigem o uso de uma biblioteca de troca de mensagens e não são eficientes com múltiplos processos e sim com a divisão de um processo pai em partes que serão executadas de forma paralela. Cabe ao programador identificar os pontos ideais de distribuição na aplicação e colocar chamadas às funções da biblioteca de troca de mensagens para indicar ao gerenciador do *cluster* o que deve ser distribuído, sendo que o desempenho fica vinculado à habilidade do programador em identificar e programar corretamente estes pontos ideais de distribuição.

As bibliotecas para troca de mensagens mais conhecidas são a PVM (*Parallel Virtual Machine*) e a MPI (*Message Passing Interface*), ambas baseiam-se na primitiva de "enviar e receber". A PVM é a mais difundida, segundo Barreto et al., PVM é mais do que uma biblioteca de comunicação, trata-se de um software básico com suporte próprio ao gerenciamento de processos e centrado no conceito de máquina virtual. A MPI, na verdade, estabelece apenas um padrão, especificando a sintaxe e semântica de um conjunto de funções de comunicação visando atender necessidades específicas para aplicações paralelas. Possui diversas implementações, muitas de código aberto como MPICH e Open-MPI, outras comerciais como HP-MPI. O padrão MPI caminha a passos largos para ser o novo padrão da indústria de software, suportando recursos como *broadcast*, que é o envio de mensagem para todos os computadores do cluster e também *multicast*, que é o envio para apenas um grupo de máquinas, oferecendo desta forma excelente suporte à comunicação coletiva o que facilita, por exemplo, o desenvolvimento de aplicações que efetuam freqüentes operações sobre matrizes.

3. O ALGORITMO A PRIORI

O algoritmo *Apriori* (Agrawal e Srikant, 1994) é um dos mais conhecidos para a tarefa de associação, buscando encontrar os relacionamentos significativos ou padrões freqüentes

em conjuntos de dados conforme os limites de suporte e confiança informados. Em um relacionamento "X=>Y", onde "X" é antecedente e "Y" é conseqüente, o suporte seria o percentual calculado sobre a freqüência com que ocorre "X=>Y" perante a base de dados, confiança é uma medida calculada pelo percentual de "X" que possui "Y", sendo um indicador de força do conjunto, mas que deve ser analisado com muito cuidado sempre considerando o valor do suporte, porque altos percentuais de confiança com baixos valores de suporte fatalmente podem levar a tomada de decisões equivocadas.

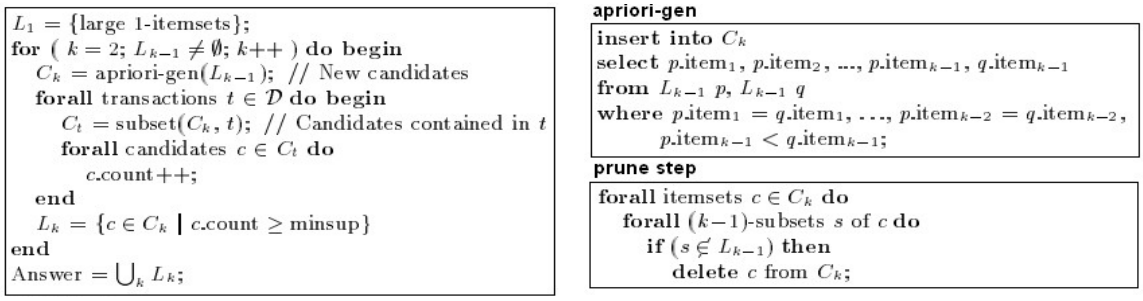

Figura 1. Algoritmo *Apriori* proposto por Agrawal & Srikant (1994).

Várias propostas para otimização do algoritmo têm sido apresentadas. Por exemplo, de Amo (2004) propõe a otimização através do uso de uma árvore *hash* (ou de espalhamento) onde as folhas armazenam os conjuntos candidatos e os nós intermediários, inclusive o raiz, armazenam tabelas *hash* contendo a chave gerada e o ponteiro para o suposto local onde o conjunto candidato estaria armazenado.

4. IMPLEMENTAÇÃO DE UM MODELO DISTRIBUÍDO DE MINERAÇÃO

A ferramenta utiliza um mecanismo para processamento paralelo baseado no conceito de *cluster de workstation* e de *client/server*, estando a aplicação de mineração dividida em duas partes: uma parte "servidora" para gerenciamento da mineração e da comunicação com os nós da rede e uma parte "cliente" que trabalha nos nós da rede, sendo responsável por receber o lote de dados e a tarefa de mineração a ser realizada. Após concluir o trabalho de minerar combinações com determinado número de elementos, envia os resultados novamente para a parte servidora, podendo receber uma nova tarefa.

A troca de mensagens baseia-se na primitiva de "enviar e receber" assim como nas bibliotecas PVM e MPI, mas utilizando um protocolo bem mais simplificado, conforme os propósitos de mineração distribuída específicos desta aplicação. O algoritmo de mineração está codificado de forma que a parte de gerenciamento da mineração ficou no servidor e a parte passível de execução paralela ficou no cliente, o processamento paralelo ocorre com as várias instâncias da aplicação cliente executando nos nós da rede, cada qual trabalhando na mineração de conjuntos freqüentes com um determinado número de elementos, conforme a distribuição de tarefas feita pelo servidor, que é responsável pela posterior coleta dos resultados.

O módulo cliente conecta-se ao gerenciador para obter os dados e a tarefa a ser realizada. Ao processar a tarefa que lhe foi atribuída, o módulo cliente vai informando ao gerenciador sobre o percentual concluído e o índice de eficiência da função *hash*, utilizada para acesso de alto desempenho aos elementos da lista de conjuntos descobertos na respectiva tarefa. Ao terminar, o módulo cliente envia o resultado ao gerenciador de tarefas que responde pedindo ao cliente que finalize sua conexão e informa se há mais tarefas aguardando na fila para execução, situação na qual o cliente reconecta-se ao gerenciador para obter a nova tarefa.

A versão inicial da ferramenta foi escrita em *Object Pascal* para ambiente Microsoft Windows, sendo que o módulo gerenciador de conexões e tarefas ainda continua nesta

linguagem e o módulo cliente já está reescrito em C++, utilizando o compilador GNU GCC, gerando executáveis para Windows através do MinGW. A comunicação e troca de mensagens é baseada em TCP/IP utilizando *sockets*, a migração do gerenciador de conexões e tarefas para C++ já está sendo planejada.

Para fornecer os dados de entrada foi adotado o formato CSV (*Comma Separated Values*). Um dos motivos da escolha é a facilidade para representar e transmitir resultados de consultas em tal formato, *e.g.* uma transação de venda por linha.

Figura 2. O Gerenciador de tarefas em execução.

O módulo gerenciador exige uma porta TCP, que pode ser alterada até o momento em que o serviço seja iniciado, observe que a tarefa de mineração vai sendo distribuída conforme as conexões clientes vão entrando. O valor de suporte informado pode ser usado tanto em forma de percentual como na forma de quantidade mínima de ocorrências. No exemplo da figura-2, podemos observar que a mineração está usando o tipo de suporte *quantidade*, minerando os conjuntos que ocorreram ao menos dez vezes. A coluna eficiência refere-se ao índice de acertos da tabela *hash*, técnica utilizada para acesso randômico à lista de conjuntos freqüentes já descobertos em memória.

5. DETALHAMENTO DA MODELAGEM

O módulo gerenciador é responsável pelo controle das conexões clientes e pela distribuição de tarefas, detectando conexões abandonadas e recolocando as tarefas para execução. Também é responsável pela separação inicial da base de dados para envio ao cliente, de forma que, se a tarefa é processar os conjuntos com três elementos, não sejam enviadas ao cliente as transações de venda que contenham apenas um ou dois elementos, desnecessários à tarefa em questão.

Escrito usando técnicas de multiprocessamento com o uso de linhas de execução concorrentes (*threads*), o gerenciador aguarda a entrada de conexões e atribui a cada uma que chega sua respectiva linha de processamento. A conexão então é repassada para a linha de execução, que fica dedicada ao envio e recepção de mensagens com o computador cliente, o *Socket TCP/IP* opera em modo síncrono (*blocking mode*). Um mecanismo faz com que as linhas de execução que concluíram sua tarefa permaneçam em *cache* acelerando o processo de entrada das próximas conexões clientes.

Devido ao multiprocessamento, o módulo gerenciador naturalmente apresentará melhor desempenho em equipamentos com mais de um processador ou contendo processadores com mais de um núcleo, *e.g. Dual Core, Quad Core*. O equipamento que executa o módulo gerenciador também poderá ser usado para aumentar a capacidade de processamento do grupo, executando uma instância do cliente simultaneamente. Conforme o volume de dados sendo minerados, este procedimento poderá causar degradação no desempenho geral da mineração.

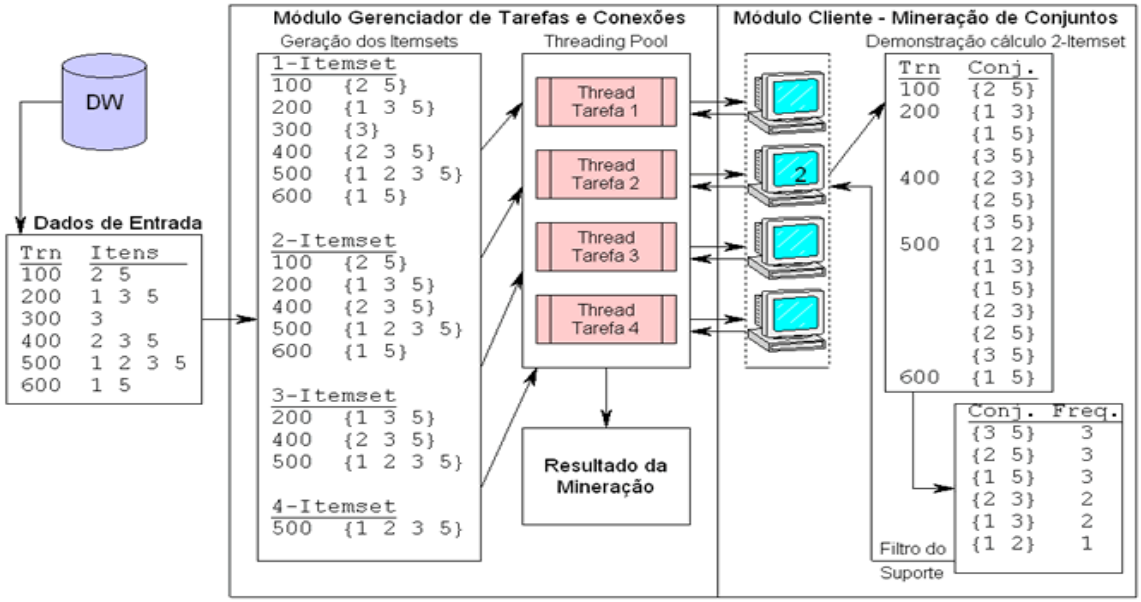

Figura 3. Diagrama de funcionamento da mineração distribuída.

O diagrama procura demonstrar as etapas da mineração, a separação dos dados de entrada antes de serem enviados ao cliente e a criação de linhas de execução no gerenciador, também um exemplo de como as regras de associação são encontradas.

O módulo cliente utiliza apenas uma linha de processamento e se comunica com o gerenciador utilizando *Sockets* de forma assíncrona (*non-blocking mode*). Durante a mineração, o cliente vai enviando mensagens com a situação atual do processamento para o gerenciador, que atualiza as informações na tela de monitoramento da mineração. A implementação do módulo cliente é fortemente baseado em *hash*, técnica utilizada para possibilitar o acesso direto (randômico) na lista não ordenada de conjuntos freqüentes descobertos. A função para a geração das chaves utiliza o método de Horner (Feofiloff, 2006) buscando melhor qualidade de dispersão, pois em grandes volumes de dados, a probabilidade de espalhamento perfeito, em que cada conjunto freqüente possui seu ponteiro exclusivo na tabela *hash* é muito pequena. Quando um candidato vai ser processado, é calculada a chave e de posse desta, tenta-se o acesso direto à lista não ordenada para verificar se o conjunto já havia sido descoberto. Em caso positivo, o contador de ocorrências é incrementado, em caso negativo, uma nova posição no final da lista não ordenada é aberta para armazenar o novo conjunto e a posição deste novo conjunto é registrada na tabela *hash* no local apropriado, conforme o valor da chave. O algoritmo possui um controle de colisões, que ocorrem quando dois conjuntos diferentes geram a mesma chave. Neste caso, a tabela *hash* aponta somente para o primeiro conjunto descoberto, sendo que os demais são vinculados a este por meio de uma lista encadeada. Desta forma, quando o algoritmo não encontrar o conjunto na primeira tentativa, provavelmente o fará logo em seguida, percorrendo a lista encadeada. Sem este mecanismo de *hash,* o algoritmo teria que percorrer sequencialmente toda a lista de conjuntos descobertos a cada nova combinação candidata que fosse gerada, tornando a mineração inviável quanto ao desempenho.

6. ESTUDO DE CASO E RESULTADOS

A ferramenta de mineração foi experimentada com uma base de dados de entrada proveniente de uma loja de departamentos do ramo varejista contendo um total de 2.487.778 registros de vendas e 63.523 produtos distintos. A mineração deste volume de dados gera uma

quantidade expressiva de combinações (conjuntos candidatos) a serem computados, é um processamento bastante pesado.

O modelo de distribuição implementado permite que a ferramenta seja executada de várias formas diferentes, uma possibilidade é executar o gerenciador e um cliente no mesmo equipamento monoprocessado fazendo com que não haja distribuição, outra é utilizar um equipamento multiprocessado e executar o gerenciador e várias instâncias do módulo cliente, que resultaria em uma distribuição dos processos cliente entre os processadores ficando limitado pela memória compartilhada do sistema e uma terceira possibilidade, que seria utilizar diversos equipamentos, um executando o gerenciador e os demais executando o módulo cliente. Esta última possibilidade foi a que revelou o melhor desempenho dentre as três.

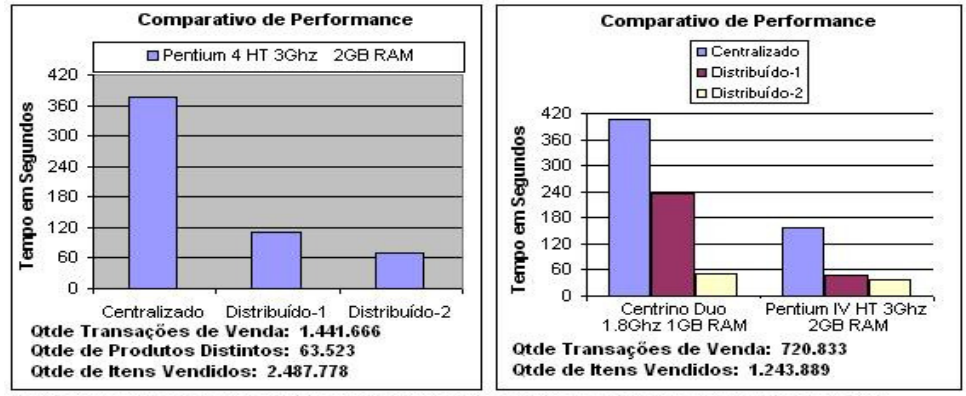

Figura 4. Comparativo de desempenho usando mineração distribuída.

O quadro de resultado apresenta apenas uma pequena amostra do resultado da mineração, revelando os conjuntos freqüentes em ordem do grau de interesse que é representado pelo número de ocorrências do conjunto. Foram adicionados o título e os nomes dos produtos para melhor visualização do resultado.

```
Prod-1 Prod-2 Qtd  Nome-1                    Nome-2
------ ------ ---- ------------------------- ---------------------------
151100,151595,2691,IVOMEC 01 DOSE,           SERINGA DESCARTAVEL 3ML
150338,151595,1198,EUROMUNI V10,             SERINGA DESCARTAVEL 3ML
000079,000080,923, FACA INOX   BELIZE,       GARFO INOX  BELIZE
184088,184683,797, SUVINIL TINTA OLEO 3.6LT, SOLVENTE AGUARRAZ 900ML

Prod-1 Prod-2 Prod-3 Qtd Nome-1                    Nome-2                    Nome-3
------ ------ ------ --- ------------------------- ------------------------- -------------------------
000079,000080,000081,453,FACA INOX BELIZE,         GARFO INOX BELIZE,        COLHER INOX BELIZE
182773,186286,186288,314,CABO COAXIAL 75 OHMS,     BALOOM EXTERNO,           CONECTOR C/ ROSCA 2 PC
182581,182757,182782,220,PINO LORENZETTI (MACHO),  PINO LORENZETTI (FEMEA),  FIO PARALELO 2X2.5 MM
181884,181888,181890,195,MT CHAMINE DE ZINCO 11.5C,CURVA P/CHAMINE ZINCO11.5,TOCO CHAMINE ZINCO 11.5CM

Prod-1 Prod-2 Prod-3 Prod-4 Qtd Nome-1                          Nome-2                    Nome-3
------ ------ ------ ------ --- ------------------------------- ------------------------- -----------------
181884,181888,181890,183963,142,MT CHAMINE DE ZINCO 11.5C,      CURVA P/CHAMINE ZINCO11.5,TOCO CHAMINE ZINC
000079,000080,000081,000082,68, FACA INOX BELIZE,               GARFO INOX BELIZE,        COLHER INOX BELIZ
181453,181454,181463,181464,63, ESPALHADOR GRANDE DAKO,         BACIA GRANDE DAKO,        ESPALHADOR PEQUEN
182773,186286,186287,186288,60, CABO COAXIAL 75 OHMS,           BALOOM EXTERNO,           BALOOM INTERNO,
```

Figura 5. Trechos do resultado da mineração de conjuntos freqüentes.

7. CONCLUSÃO E PERSPECTIVAS FUTURAS

A ferramenta apresentada tem como vantagem a simplicidade com que permite minerar grandes volumes de dados através do uso de processamento paralelo, não exigindo a montagem de um *cluster* dedicado para isso, mas podendo a qualquer tempo utilizar-se dos equipamentos de uso diário em uma rede local para formar um conjunto capaz de processar e

concluir com sucesso a árdua tarefa de experimentar e associar uma grande quantidade de conjuntos candidatos (combinações) gerados neste tipo de mineração de dados, mas também possui as limitações e desvantagens citadas durante a abordagem da modalidade c*luster de workstation*.

A partir dos resultados obtidos por meio desta ferramenta, é possível fazer uma análise humana com base na avaliação dos conjuntos com maior grau de interesse e aproveitar estas informações para adotar estratégias eficientes visando o incremento das vendas, podendo auxiliar na alocação física de mercadorias em exposição, bem como na definição de campanhas e promoções.

Gestores podem tirar proveito das informações fornecidas pela mineração, que pode indicar inclusive políticas de preços, como por exemplo, analisando as duas primeiras linhas com dois elementos podemos observar um produto em comum que é o código 151595 (Seringa). Um gerente pode determinar preços competitivos de vacinas (códigos 151100 e 150338) e margens amplas nas seringas, pois os clientes quase sempre decidem a compra pelo preço das vacinas e não pelo da seringa. Outro exemplo seria a primeira linha com três elementos (Faca, Garfo e Colher vendidos em avulso), são itens de baixo custo com possibilidade de alto lucro. Este conjunto, pela freqüência elevada, merece um local de destaque na alocação física e a aproximação de outras miudezas correlatas. De fato, a mineração forneceu informações estratégicas que podem ser utilizadas para tornar a empresa mais competitiva.

As próximas versões da ferramenta deverão contemplar avanços em termos de portabilidade, com a conclusão da migração do código-fonte de *Object Pascal* para C++ baseado-se no compilador GNU GCC, de modo a gerar executáveis para diversas plataformas. Também está em estudo aumentar a granularidade do modelo distribuído através de um maior particionamento da mineração, o que traria muitos benefícios no processamento de conjuntos candidatos compostos por um grande número de elementos combinados, além do desenvolvimento e inclusão de um escalonador para melhor distribuição de tarefas, principalmente quando os computadores do grupo tiverem diferentes recursos de memória e capacidade de processamento.

8. REFERÊNCIAS

AGRAWAL, Rakesh; SRIKANT, Ramakrishnan. Fast Algorithms for Mining Association Rules. Proceedings of the 20th International Conference on Very Large Databases: Santiago – Chile, 1994.

BARRETO, Marcos; ÁVILA, Rafael; OLIVEIRA, Fábio. Execução de Aplicações em Ambientes Concorrentes. Instituto de Informática da Universidade Federal do Rio Grande do Sul - UFRGS.

BUENO, André Duarte. Introdução ao Processamento Paralelo e ao Uso de Clusters, Parte I: Filosofia. Laboratório de Meios Porosos e Propriedades Termofísicas LMPT – UFSC, 2002.
De AMO, Sandra. Técnicas de Mineração de Dados. Faculdade de Computação da Universidade Federal de Uberlândia, 2004.

FEOFILOFF, P. Tabelas de dispersão. Material disponível em: www.ime.usp.br/~pf/mac0122-2002/aulas/hashing.html. Data da consulta: 06/2007

KIMBALL, Ralph; ROSS Margy. The Data Warehouse Toolkit – Second Edition.
USA Wiley Computer Publishing, 2002.

MORRISON, Richard S. Cluster Computing - Architectures, Operating Systems, Parallel Processing & Programming Languages. University of Technology – Sydney, 2003.

SILVA, Marcelino P. Santos. Mineração de dados: Conceitos, Aplicações e Experimentos com Weka. Universidade do Estado do Rio Grande do Norte (UERN).

WITTEN, Ian H.; FRANK, Eibe. Data Mining: Practical Machine Learning Tools and Techniques. 2nd Edition, Morgan Kaufmann. San Francisco, 2005.

Wikipédia - Cluster - http://pt.wikipedia.org/wiki/Cluster. Data da consulta: 07/2007