UNILASALLE
CENTRO UNIVERSITÁRIO LA SALLE

Curso de Bacharelado em Ciência da Computação

RAFAEL FRÖHLICH DUTRA

PUBLICAÇÃO E GERENCIAMENTO DE INFORMAÇÕES DE MONITORAMENTO DE RECURSOS EM AMBIENTE DE GRADE

CANOAS, 2008

RAFAEL FRÖHLICH DUTRA

PUBLICAÇÃO E GERENCIAMENTO DE INFORMAÇÕES DE MONITORAMENTO DE RECURSOS EM AMBIENTE DE GRADE

Monografia apresentada ao Curso de Ciência da Computação do Centro Universitário La Salle, como parte dos requisitos para obtenção do título de Bacharel em Ciência da Computação, sob a orientação da Profa. DSc. Patrícia Kayser Vargas Mangan.

CANOAS, 2008

RESUMO

Este trabalho apresenta o contexto de monitoramento de recursos em ambiente de grade, descrevendo a arquitetura implementada pelo monitor MoonGrid, analisando as questões de modelagem de acordo com especificações sugeridas pelo Open Grid Forum e propondo o desenvolvimento de um módulo para análise dos dados coletados pelos sensores do monitor. O MoonGrid, fruto do trabalho de conclusão de curso de Daniel da Trindade Lemos, foi criado com a intenção de prover monitoramento de hardware e software e pode ser utilizado por um sistema gerenciador de aplicações em grade. Uma das questões sugeridas como trabalho futuro diz respeito à publicação e gerenciamento das informações coletadas pelos sensores de monitoramento. Este texto apresenta os esforços acerca desta questão, incluíndo a proposta e desenvolvimento de um módulo que implementa Mineração de Dados para descoberta de conhecimento por meio de Regras de Associação. Os resultados incluem a descrição das regras geradas durante os testes e análise de desempenho do processo de geração.

Palavras-chave: computação em grade, monitoramento de recursos, mineração de dados.

ABSTRACT

This text presents the context of grid resource monitoring, describing the architecture created by the MoonGrid monitor, analysing the modelling questions according to the Open Grid Forum specifications, and presenting the development of a new data analysis module. The MoonGrid, that was born as part of Daniel Trindade Lemos's monograph text, was created to provide hardware and software monitoring and can be used by a grid application management system. One of the problems suggested as a future work is the publishing and management of the data collected by the monitoring sensors. This text presents the efforts about this subject, including a proposal and deployment of a new module that uses Data Mining to get Knowledge Discovery and generate Association Rules. The results include the description of the rules generated during the tests and the performance evaluation about the generation process.

Keywords: grid computing, resource monitoring, data mining.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

Banrisul: Banco do Estado do Rio Grande do Sul
EDG: European Data Grid
ER: Entidade Relacionamento
GMA: Grid Monitoring Architecture
GRAND: Grid Robust Application Deployment
MoonGrid: Monitoring on Grid
R-GMA: Relational Grid Monitoring Architecture
SQL: Structured Query Language
T-SQL: Transact SQL

SUMÁRIO

1 INTRODUÇÃO

A possibilidade de se ter máquinas entre instituições distintas, disponibilizando seus recursos computacionais para solucionar os problemas de outros usuários e instituições por meio de uma rede integrada, fez surgir o conceito Computação em Grade (*grid computing*) [1]. Ambientes de computação em grade têm como característica principal a heterogeneidade de recursos geograficamente distribuídos. Questões como disponibilidade, detecção de falhas, análise de desempenho, controle de submissão e execução de tarefas tornam o monitoramento de recursos essenciais para a natureza do ambiente.

Dos principais pontos a serem pesquisados no contexto de ambientes computacionais de grade, o tratamento dado às informações coletadas pelo monitoramento dos recursos é, de fato, o que fará do ambiente um sistema coerente e otimizado para utilização de seus recursos. As possibilidades diante da enorme quantidade de informações geradas devem ser desenvolvidas de modo que se consiga extrair informação necessária para a tomada de decisões, tanto administrativas em relação à grade, quanto técnicas em relação à distribuição de tarefas pelo sistema gerenciador. A grande quantidade de dados de monitoramento disponibilizadas por um ambiente de grade torna fundamental a caracterização destes dados em informação inteligente, que possa ser utilizada de maneira prática e efetiva dentro do ambiente.

O MoonGrid [2] é um monitor de recursos que foi desenvolvido no contexto do projeto GRAND [3], e que provê monitoramento de hardware e software. O MoonGrid tira proveito dos aspectos estáticos e dinâmicos das informações para otimizar a coleta e envio das informações. Ele foi desenvolvido para ser utilizado por um sistema gerenciador de aplicações em ambiente em grade, como o GRAND, mas também é genérico o suficiente para ser usado por outras ferramentas

administrativas. Este sistema é um exemplo de ferramenta capaz de obter um grande conjunto de informações sobre a grade, mas que não dispõe de mecanismos para gerar informações de mais alto nível.

Diante de todas as questões críticas apresentadas por um ambiente computacional de grade e das principais questões levantadas pelo sistema MoonGrid, o presente trabalho têm como foco principal abordar o gerenciamento das informações de monitoramento coletadas e publicadas pelos sensores de monitoramento do MoonGrid.

Este texto apresenta estudos na área de publicação e gerenciamento de informações para ambientes em grade, bem como a proposta de um modelo para extração de informação inteligente dos dados coletados pelo sistema MoonGrid. O objetivo é o desenvolvimento de um módulo de gerenciamento dos dados que forneça informação de alto nível, priorizando as informações de utilização de software, pois estas são o diferencial apresentado pelo sistema MoonGrid.

Assim, um dos objetivos deste trabalho é estudar modelos e técnicas para tratamento de grande quantidade de informação aplicado ao monitoramento de recursos, a fim de determinar quais as melhores alternativas diante das características do sistema MoonGrid, visando auxiliar escalonadores de recursos e também administradores do ambiente.

Este trabalho é classificado como um estudo de caso de natureza aplicada, pois testa modelos de extração de informação de monitoramento para um ambiente específico.

Os resultados dos estudos sobre adequação do modelo MoonGrid ao padrão GMA [5] mostraram que a possível convergência seria interessante para uma segunda versão do sistema monitor, uma vez que os serviços básicos propostos pelo padrão são oferecidos pelo MoonGrid, e as divergências referentes à publicação e gerenciamento de informação não justificam alterações na estrutura do modelo atual.

Outra importante contribuição deste trabalho foi a proposta e desenvolvimento de um Módulo de Análise que implementa *Data Mining* para extração de informação de alto nível. A avaliação da proposta foi feita a partir de um conjunto de dados obtidos junto ao Banrisul, onde o monitor MoonGrid está atualmente instalado.

O restante do texto apresenta-se organizado do seguinte modo. Inicialmente apresenta-se o Capítulo 2 sobre Monitoramento em Grade, onde é explicado o

contexto do trabalho e detalhadas as arquiteturas do monitor MoonGrid e do padrão proposto pelo *Open Grid Forum* [4]. Em seguida, o Capítulo 3 apresenta os estudos sobre *Data Mining* relativos ao modelo proposto e implementação. Depois, o Capítulo 4 demonstra os experimentos efetuados e analisa os resultados obtidos. Por fim, o Capítulo 5 conclui este trabalho e relaciona as possibilidades de trabalhos futuros.

2 MONITORAMENTO DE RECURSOS EM GRADE

Atualmente é possível a criação de ambientes de rede onde máquinas, heterogêneas em suas características e geograficamente distribuídas, se conectam através de uma estrutura de hardware e software concebida por sistemas distribuídos ou grades computacionais [4].

Segundo Mattos [11] uma grade computacional é uma infra-estrutura de hardware e software que provê acesso seguro, de forma distribuída a um baixo custo.

Outra definição para grade é apresentada em Krauter *et al.* [12]: “uma grade é um sistema computacional de rede que pode escalar para ambientes do tamanho da Internet, com máquinas distribuídas através de múltiplas organizações e domínios administrativos. Neste contexto, um sistema computacional de rede distribuído é um computador virtual formado por um conjunto de máquinas heterogêneas, ligadas por uma rede, que concordam em compartilhar seus recursos locais com os outros.”

Neste capítulo é descrito a arquitetura do modelo implementando pelo monitor MoonGrid, juntamente com a validação desta arquitetura de acordo com o GMA (*Grid Monitoring Architecture*) [5], modelo proposto pelo *Open Grid Forum* [4]. O *Open Grid Forum* é uma comunidade de usuários e desenvolvedores liderando um esforço global de padronização para computação em grade. Também é descrito neste capítulo a implementação relacional proposta sobre o padrão GMA, o R-GMA *(Relational Grid Monitoring Architecture*).

2.1 MoonGrid

A idéia concebida pelo monitor de recursos MoonGrid propõe resolver as deficiências de ferramentas tradicionais de monitoramento quanto à escalabilidade e

obtenção de características de software dos nós da grade [2]. A seguir são descritos os principais pontos da arquitetura e uma análise sobre o modelo implementado pelo monitor.

2.1.1 Modelo de Monitoramento

O ambiente a ser monitorado consiste em uma grade computacional heterogênea e com estrutura de nós possivelmente dinâmica. Todos os computadores de um nó são capazes de se comunicar com um computador comum que centraliza as informações coletadas pelos sensores. Esse computador é responsável pelo envio das informações ao servidor, que as armazena no repositório. Em cada máquina um sensor é instalado e no instante que esta é ligada o sensor faz a leitura dos recursos, software e hardware, para mais tarde enviar os dados ao centralizador.

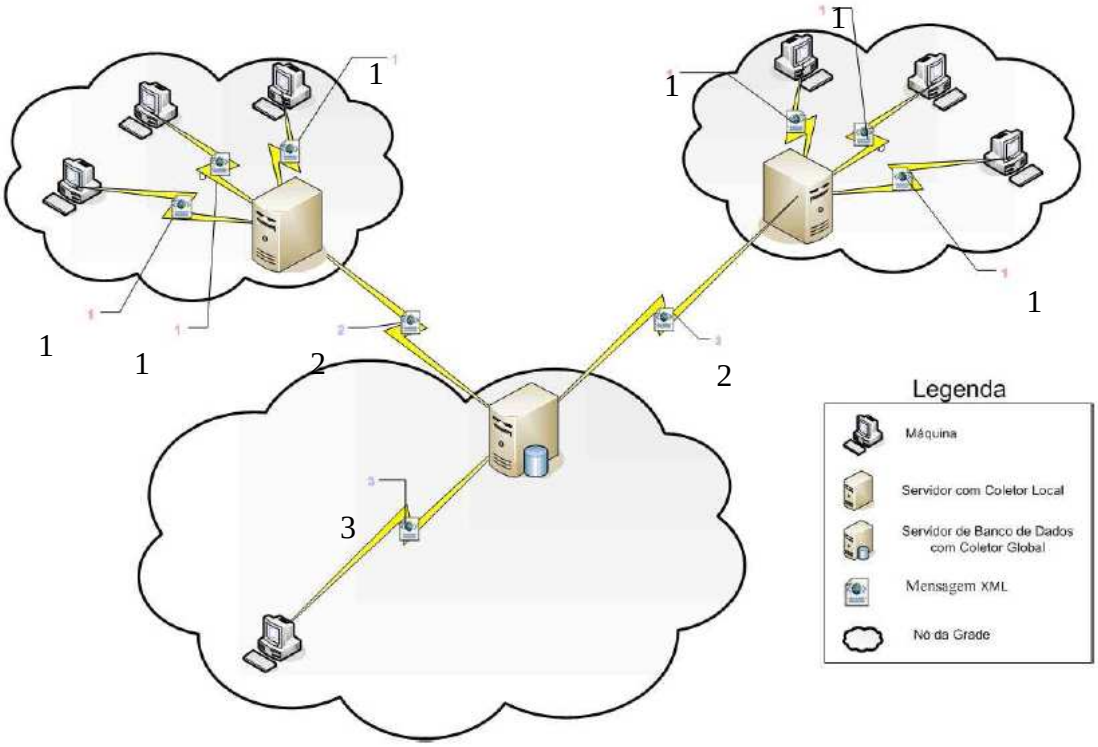


Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.1: Visão Geral do modelo MoonGrid
Fonte: Lemos, D. da T., 2006 [2]

Conforme mostra a Figura 2.1 existem três componentes responsáveis pela coleta e armazenagem das informações: Sensor, Coletor Local e Coletor Global. O Sensor faz a coleta dos dados localmente e o envio destes para o Coletor Local. O Coletor Local centraliza os dados do nó da grade (existe um Coletor Local por cluster ou rede local) e o Coletor Global é quem têm a visão geral de todos os recursos da grade. A freqüência de envio dos dados aos Coletores Locais (comunicação identificada com número “1” na figura) e Globais (comunicação identificada com número “2”) é parametrizada pelo gerenciador da grade no momento da concepção da mesma. A partir do Coletor Global, qualquer máquina poderia solicitar estes dados (comunicação identificada com número “3”). Aqui mantem-se o foco do presente trabalho.

2.2 Arquitetura de Monitoramento em Grade (GMA)

Esta seção apresenta o núcleo (*core*) da arquitetura proposta pelo *Open Grid Forum* [4], através do *Performance Working Group* [5], em um modelo de alto-nível para comunicação entre diferentes tipos de componentes. A descrição e análise a seguir não dizem respeito à criação e gerenciamento de componentes nem de como devem ser publicadas informações sobre os recursos, pois o *Open Grid Forum* ainda não publicou documentos que contribuem com este grau de detalhe.

2.2.1 Arquitetura e Terminologia

A arquitetura GMA consiste na composição de três tipos de componentes, mostrados na Figura 2.2:

- Serviço Diretório: provê publicação e descoberta de informação.
- Produtor: produz e disponibiliza dados sobre o recurso.
- Consumidor: recebe dados sobre o recurso.

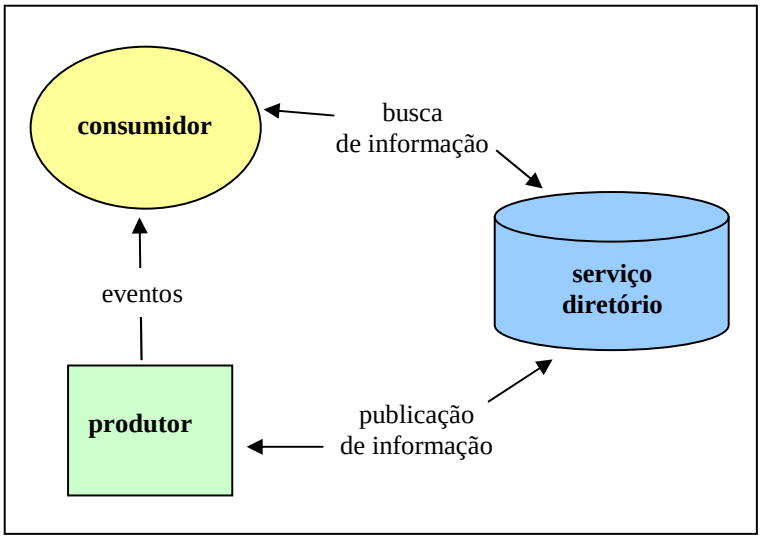


Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.2: Componentes da Arquitetura GMA
Fonte: traduzido de [5]

A arquitetura GMA é projetada para lidar com transmissão de dados de desempenho através de eventos *timestamped*. Um evento é uma coleção de dados com uma estrutura específica definida por um esquema de evento. Dados de desempenho são sempre enviados diretamente do Produtor para o Consumidor.

2.2.2 Componentes

Esta seção descreve as definições e funcionalidades dos componentes Consumidor, Produtor e Serviço Diretório.

- **serviço diretório**: para descobrir e descrever dados de desempenho em Grade deve haver um Serviço Diretório distribuído que disponibiliza publicação e busca. O Serviço Diretório GMA contém informação sobre Produtores e Consumidores. Quando Produtores e Consumidores publicam sua existência no Serviço Diretório eles especificam qual tipo de evento eles produzem ou consomem. Esta informação permite que outros Produtores e Consumidores descubram quais dados e eventos estão atualmente disponíveis, suas características e o destino ou origem que irá aceitar ou produzir cada tipo de dado. O Serviço

Diretório não é responsável por armazenar dados produzidos, mas sim por publicar informação sobre quais eventos estão disponíveis.

- **produtor**: é qualquer componente que envie eventos ao Consumidor.
- **consumidor**: é qualquer componente que receba dados de um Produtor.

2.3 R-GMA (*Relational Grid Monitoring Architecture*)

O R-GMA é uma implementação relacional do GMA, desenvolvido pelo EDG (*European Data Grid*) [7], com o intuito de prover ao modelo a flexibilidade da estrutura relacional. Como visto na seção anterior, a arquitetura GMA consiste na junção de três componentes: Produtor, Consumidor e Serviço Diretório, que no R-GMA é referido como Registro.

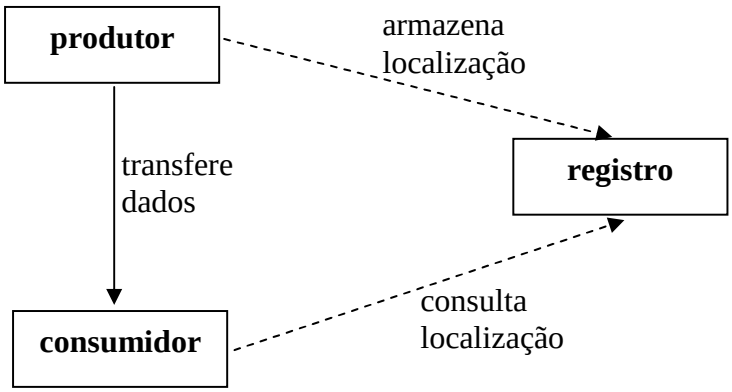


Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.3: R-GMA – visão geral
Fonte: traduzido de [7]

No GMA Produtores registram a si mesmos no Registro e descrevem o tipo e estrutura de informação que desejam tornar disponível para a Grade. Consumidores podem consultar o Registro para descobrir o tipo de informação disponível e localizar Produtores que dispõem destes dados.

O GMA também define o Registro de Consumidores. A principal razão para registrar a existência de Consumidores é que o Registro pode notificá-los sobre mudanças no conjunto de Produtores interessados neles.

O R-GMA cria a impressão de que temos um banco de dados relacional distribuído. Entretanto é preciso ficar claro que esta visão é uma abstração sobre a

estrutura GMA. Todos os produtores de informação são independentes. O modelo é relacional, no sentido de que Produtores anunciam o que querem publicar através de um comando SQL CREATE TABLE e publicam com um SQL INSERT, Consumidores usam um SQL SELECT para coletar a informação de que precisam.

2.4 Conclusões sobre análise de convergência para o padrão

Após estes estudos e pesquisas nas bases do *Open Grid Forum* [4] e *European Data Grid [7]*, percebe-se que as principais características que diferem a implementação do MoonGrid da arquitetura proposta pelo grupo de performance do *Open Grid Forum*, dizem respeito ao próprio conceito inicial que motivou a construção do MoonGrid bem como dos objetivos incentivados pelo GMA. O monitor MoonGrid baseou seu projeto na carência de maior controle sobre os recursos físicos e lógicos apresentada pelas atuais ferramentas e padrões para monitoramento de recursos em grade. Assim como o GMA, os demais projetos existentes não definem estruturas específicas para os dados que estão sendo monitorados. Na realidade, a maioria define seu escopo principal de recurso como sendo uma máquina agregada a nós da grade, não se preocupando com o perfil deste recurso e, deste modo, também não detalhando individualmente nenhum recurso.

Como as premissas principais do projeto MoonGrid são justamente o detalhamento dos recursos de hardware e software presentes na grade, permitindo assim o maior controle individual dos recursos, verifica-se que a implementação do padrão GMA sobre a arquitetura do MoonGrid não contribuiria hoje com o projeto de maneira efetiva.

No entanto, para uma proposta de trabalho futuro, a arquitetura do sistema MoonGrid permite a convergência tanto para o padrão GMA quanto para o R-GMA, sendo este último mais interessante devido às características de banco de dados relacional. Isso poderia ser feito considerando-se o seguinte mapeamento:

Sensor → Produtor

Coletor Local → Consumidor

Novo módulo para desacoplar Sensor e Coletor → Serviço Diretório/Registro

Coletor Local → Produtor

Coletor Global → Consumidor

Registry do WS → Serviço Diretório/Registro

2.5 Considerações finais

Foram descritos até aqui os principais itens de modelagem e arquitetura do ambiente MoonGrid, validados de acordo com as práticas sugeridas pela comunidade de grade. Conforme estes primeiros resultados, o foco agora se atém ao segundo problema.

Os próximos capítulos apresentam detalhes do trabalho relativo a questão do tratamento dado as informações coletadas pelos sensores de monitoramento.

3 MODELO PARA GERENCIAMENTO DE INFORMAÇÃO DE MONITORAMENTO DE RECURSOS

Com a intenção de fornecer a necessária infra-estrutura para um sistema em grade vários projetos e ferramentas têm sido desenvolvidos, como Globus Toolkit [8], Condor [19] ou Ganglia [20]. Com relação à visualização das informações sobre os recursos da grade, estes projetos focam basicamente na atividade corrente dos recursos (máquinas) e na disponibilidade dos mesmos no instante em que se visualizam as informações. A maioria destes sistemas também não apresenta nenhum tratamento para informações sobre dados históricos da grade, o que demonstra ainda uma lacuna importante a ser observada, uma vez que a análise deste tipo de informação tende a contribuir muito para aquisição de conhecimento em grandes ambientes computacionais [15].

Como o monitor MoonGrid pode ser utilizado por um sistema gerenciador de aplicações em grade, as informações de estado corrente que disponibiliza podem servir para a tomada de decisão quanto ao escalonamento e distribuição de tarefas. Porém, vários trabalhos apontam que o uso de informações do histórico como uma forma de detectar tendência de utilização dos recursos pode ser uma forma de obter um escalonamento melhor. De acordo com Wilhelms Damasio [16], através da coleta de um histórico relativo de certas características dos processos, busca-se identificar numericamente a qual categoria o processo se enquadra, tendo como indicador fatias de tempo e troca de contexto. O MoonGrid atualmente armazena as informações, mas não disponibiliza um histórico organizado.

Outro importante ponto a ser observado, diz respeito ao atual foco do monitor MoonGrid, cuja versão corrente é utilizada pelo Banrisul para análise de informações de acesso a software de maneira estatística. As consultas são feitas sob demanda, por profissionais específicos que recuperam as informações utilizando script SQL. O

MoonGrid, que inicialmente foi concebido para que o ambiente de grade recuperasse informações de acesso a software, conseguiu consolidar sua aplicabilidade. No entanto, não fazia parte do escopo do projeto criar mecanismos de controle ou inferência sobre as informações coletadas.

Assim, a principal motivação para o presente trabalho é dispor um modelo onde se possam analisar individualmente características de cada recurso e seu histórico no ambiente, aproveitando o passado de utilização do sistema monitor desde sua criação até hoje. Para atingir este objetivo, optou-se por aplicar técnicas de *Data Mining*.

Este capítulo apresenta informações sobre os experimentos com o algoritmo de *Data Mining* para descoberta de regras de associação, denominado *Apriori* [13]. Este algoritmo foi escolhido por ser um dos mais difundidos e com documentação de mais fácil acesso atualmente, atingindo resultados sempre interessantes [13] [14].

3.1 Mineração de Dados – Descoberta de regras de associação

Regra de associação é uma forma de representação de conhecimento que, através da generalização e classificação, permite o prognóstico relacionando alguns atributos que não somente a classe em questão. Segundo Freitas & Lavington [9], na sua forma original, regras de associação trabalham com um tipo especial de dados, chamados dados de *basket* (analogia a cesta de supermercado), onde uma tupla corresponde a uma transação e possui um conjunto de atributos chamados itens.

A motivação para a implementação de um algoritmo que descubra regras de associação [13] diz respeito à grande variedade de aplicações possíveis para estas regras. Questões relacionadas ao perfil e características de acesso normalmente são quantificadas através da análise direta das informações como horário e tempo de utilização. A junção de um algoritmo de *Data Mining* à arquitetura do sistema MoonGrid também se mostra interessante pela necessidade de alto desempenho na análise de grandes quantidades de informação. Por ser um campo de pesquisa multidisciplinar, que envolve estatística, aprendizado, banco de dados, sistemas especialistas e técnicas de visualização de informação, desempenho é foco constante em projetos de *Data Mining* [17].

A mineração de dados por regras de associação é realizada por meio de técnicas automatizadas para análise de grande quantidade de dados, procurando extrair informações que estejam implícitas, previamente desconhecidas, mas potencialmente úteis.

Uma regra de associação é um relacionamento na forma

X → Y

onde **X** e **Y** são conjuntos de itens e **X ∩ Y = Ø**.

Muitas vezes o número de regras geradas é muito grande e a maioria delas não contém conhecimento relevante para o conjunto de dados. Com a intenção de determinar quais as melhores regras diante da informação apresentada, dois parâmetros de medida foram introduzidos aos algoritmos de descoberta de regras de associação:

- suporte: determina o número de instâncias à que a regra se aplica, é definido como a razão do número de tuplas que satisfaçam tanto **X** quanto **Y** sobre o número total de tuplas, isto é, **Suporte = |X U Y| / N**, onde **N** corresponde ao número total de tuplas;
- confiança: expressa o número de instâncias que podem ser corretamente previstas em proporção a todas as instâncias a que a regra se aplica, é definido como a razão do número de tuplas que satisfaçam tanto **X** quanto **Y** sobre o número de tuplas que satisfazem **X**, isto é, **Confiança = |X U Y| / |X|**.

O que se pretende com o este trabalho é testar um modelo de descoberta de regras de associação bem como analisar a existência de correlação entre os dados publicados pelos sensores do monitor MoonGrid, para possibilitar análise detalhada sobre as informações.

A determinação de valores adequados para os parâmetros suporte e confiança, via de regra, é um dos grandes desafios para viabilizar a obtenção de resultados úteis. Para uma maior corretude na escolha e avaliação de regras de associação alguns índices denominados Medidas de Interesse [18] podem ser verificados. Neste trabalho, a metodologia utilizada para configurar estes valores foi avaliação da Medida de Interesse Objetiva *Lift* [18], que é dada pela razão entre a Confiança da Regra e o Suporte em relação ao Sucessor da Regra.

Lift (X → Y) = Conf (X → Y) / Sup (Y)

3.2 Algoritmo Apriori

Dentre os algoritmos existentes para descoberta de regras de associação, um dos mais difundidos e implementados em diversas ferramentas de *Data Mining* é o algoritmo *Apriori* [13]. Conforme detalhado por Agrawal *et al*. [13] e também verificado por Boscoroli *et. al.* [6], o algoritmo *Apriori* é bastante eficaz em relação aos outros algoritmos conhecidos. A escolha deste algoritmo para a implementação do Módulo de Análise tem como principais justificativas o seu bom desempenho e documentação de fácil acesso.

Este algoritmo recebe como parâmetro um conjunto de transações **T**, um valor percentual **S** como o suporte e um valor percentual **C** como confiança. É gerado então um conjunto de regras no formato

X → Y [suporte, confiança]

onde o conjunto **X** é chamado de antecedente da regra e o conjunto **Y** é chamado de conseqüente.

O algoritmo *Apriori* é dividido em duas partes. Na primeira parte (linha 3 na Figura 3.1) são selecionados todos os subconjuntos de **T** que podem ser utilizados em alguma regra, ou seja, que contenham o suporte acima do suporte mínimo **S**. Tais conjuntos são chamados de conjuntos de itens freqüentes (*frequent itemsets*).

A segunda parte do algoritmo (linhas 4 a 9 na Figura 3.1) faz a composição das regras a partir dos subconjuntos gerados inicialmente, combinando todos os elementos selecionados e observando a ocorrência destes sob o conjunto total. Todas as possíveis regras candidatas são geradas e testadas quanto à confiança mínima **C**. Uma regra candidata é gerada extraindo-se um subconjunto de itens do conjunto de itens freqüentes para ser o antecedente da regra e usando-se os itens restantes no conjunto de itens freqüentes para ser o conseqüente da regra. Somente as regras candidatas com confiança maior ou igual à confiança mínima **C** especificada pelo usuário são incluídas na saída do algoritmo.

```
1)  L1 = {large 1-itemsets};
2)  for ( k = 2; Lk-1 ≠ ∅; k++ ) do begin
3)     Ck = apriori-gen(Lk-1);  // New candidates
4)     forall transactions t ∈ D do begin
5)        Ct = subset(Ck, t);  // Candidates contained in t
6)        forall candidates c ∈ Ct do
7)           c.count++;
8)     end
9)     Lk = {c ∈ Ck | c.count ≥ minsup}
10) end
```

Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.1: Pseudocódigo do algoritmo Apriori
Fonte: Agrawal & Srikant [13]

A função **apriori-gen** pode ser considerada como a seleção a cada iteração de novos candidatos, respeitando suporte e confiança em relação à transação.

Adicionalmente, pode-se incorporar a um algoritmo híbrido a verificação, a cada iteração, de co-relação de outros atributos que não somente as colunas que compõe a transação, mas inclusive de entidades relacionadas. Esta característica é importante, pois determina maior acurácia para dados cuja estrutura não esteja diretamente associada a um registro do tipo transação.

3.3 Implementação e testes

Utilizando as mesmas ferramentas utilizadas para a criação do monitor MoonGrid e concebido para se integrar ao sistema de gerenciamento, o Módulo de Análise foi desenvolvido em C# e SQL Server, podendo conectar-se diretamente ao banco de dados do coletor global. Para o ambiente de desenvolvimento foram utilizados backups da base de dados original instalada no Banrisul. Este processo foi todo acompanhado por Daniel Lemos, que fornecia informações técnicas e de negócio.

Inicialmente o conceito “Módulo de Análise” foi adotado pela simples correspondência genérica ao propósito principal deste trabalho de pesquisa: obter conhecimento útil através de processos de análise sobre os dados do ambiente. Após os estudos, observou-se que o termo ainda condiz com a proposta final adotada, e abre espaço para muitos esforços futuros direcionados à análise ou utilização dos dados gerados para outras ações. Como por exemplo complementar o módulo desenvolvendo outra técnica de *Data Mining*.

O módulo implementa um algoritmo que tem como base o *Apriori* clássico. Além dos parâmetros de suporte e confiança, detalhados na seção 3.1 deste capítulo, o algoritmo recebe também a entidade principal e as colunas que contém a informação que se deseja analisar incidência e relacionamentos implícitos.

A entidade considerada como foco dos estudos deste trabalho é a UsoSoftware, representada na Figura 3.2, que armazena informações de acesso a softwares. O diagrama ER (entidade relacionamento) desta figura foi criado durante a análise da base de dados do MoonGrid, e respresenta o escopo funcional deste trabalho perante o total de informações armazenadas pelo monitor. O ER foi gerado a partir do próprio SQL Server, e estas tabelas foram inseridas porque armazenam dados centrais sobre o acesso a software de acordo com a modelagem atual do sistema monitor. É importante destacar que não se trata de uma limitação da aplicação o fato de não se utilizar outras entidades do conjunto total de dados do MoonGrid mas sim considera-se que estas sejam o ponto de maior interesse para descoberta de conhecimento.

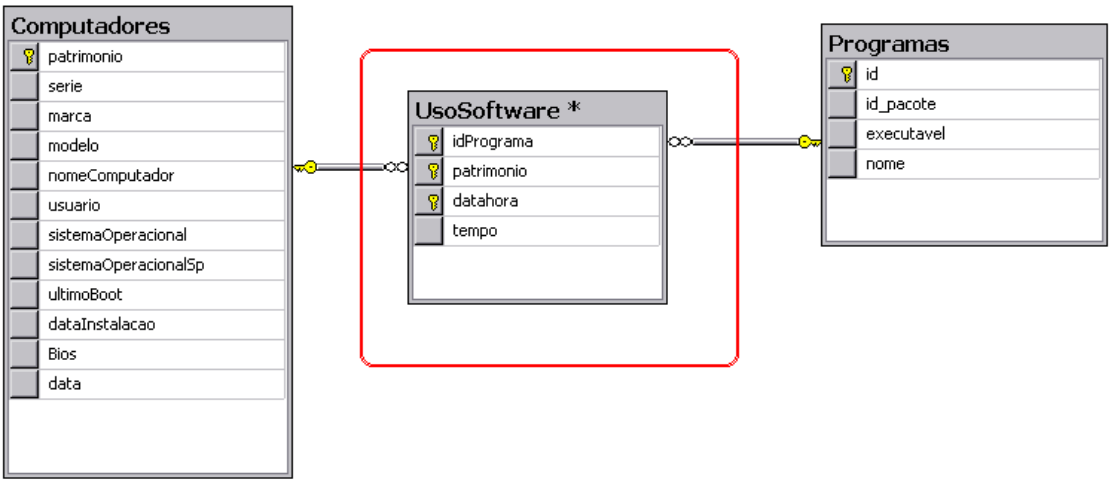


Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.2: Diagrama ER, tabela UsoSoftware
Fonte: autoria própria.

A construção do Módulo de Análise iniciou-se no início do ano corrente e por volta do segundo mês, com uma primeira versão do algoritmo implementada, começava a primeira fase de testes. Durante esta etapa surgiram importantes

questões, como fatores que determinam melhor qualidade às regras geradas e quantidade mínima de transações estatisticamente satisfatórias para geração das mesmas. Atrelado a isso, a questão do desempenho do processo de geração de regras também viria a receber atenção, pois a versão inicial do código não implementava o processo de amostragem, analisando assim muito mais informação do que a necessária para uma correta inferência. Buscando solução para estas questões, estudou-se alguns trabalhos relacionados para verificar como geralmente são tratados fatores como qualidade e quantidade de transações a serem analisadas.

Conforme indicado por Agrawal et al. [14], dos fatores que geralmente tornam ineficazes os modelos para descoberta de regras de associação, o principal diz respeito à forma de armazenagem dos dados, que quando não estruturados como transação, dificultam a execução do algoritmo e prejudicam a construção da solução. Outro ponto que geralmente é tido como complicador é o desempenho diante da quantidade de transações que se pode tomar como amostra para análise. Uma alternativa simples é formar todos os conjuntos de itens e obter seus respectivos suportes em uma única “varredura” dos dados. No entanto, esta solução não é razoável quando há grande quantidade de informação. Se existirem N itens na base de dados, haverá 2^N conjuntos de itens possíveis, e N pode facilmente ser maior que 1.000. Um método mais eficiente proposto por Agrawal *et al*. [14] é relatado a seguir:

- Estabelece-se um procedimento de estimação para determinar quais conjuntos de itens podem ser ter seu suporte contado em uma passagem pelos dados. Este procedimento atinge um meio termo entre o número de passagens pelos dados e o número de conjuntos de itens que são medidos em uma passagem;
- Incorpora-se o pré-processamento e bufferização para separar a delimitação dos *itemsets* do processo de geração de regras.

Portanto, durante a fase de desenvolvimento do Módulo de Análise passou-se a considerar estas questões como muito importantes para que este trabalho produza resultados satisfatórios.

3.3.1 *MoonGrid Mining Manager*

Esta seção descreve as principais características do modelo implementado como parte deste trabalho e solução proposta como uma aplicação que, apesar de integrada ao monitor MoonGrid, pode se conectar a qualquer modelo relacional implementado no SQL Server. Os dados descritos aqui mostram as etapas consolidadas após os Experimentos Preliminares, que são demonstrados no próximo capítulo.

Os passos para a criação de um modelo gerador de regras de associação utilizando o MoonGrid Mining Manager podem ser descritos basicamente como:

- **Etapa 1 - definição da estrutura relacional**: usuário informa as entidades principal e secundária, indicando seus elementos de relação para composição de regras. A estrutura informa ao algoritmo quais campos serão verificados em relação ao campo definido como sucessor. Se a informação desejada estiver em outra tabela, esta tabela deve ser incluída como entidade secundária. Por exemplo, considere a tabela 3.1:

Tabela **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.1: Estrutura de um Modelo MoonGrid M.Man - exemplo 1

entidade principal: UsoSoftware	
Chave:	*Patrimonio*
Sucessor da regra:	*Programa*
Input 1:	*DataHora*
Input 2:	*Tempo*

entidade relacionada 1: Computadores	
Chave:	*Patrimonio*
Input:	*Usuário*

Fonte: Autoria própria.

Neste exemplo está indicado que o Patrimônio do equipamento será considerado como único para o modelo e através da entidade secundária Computadores, pode-se verificar informações de Usuário. O parâmetro Sucessor indica sobre qual campo serão avaliadas as transações e gerados o conjunto de itens freqüentes de forma transacional. Os campos de *Input* podem vir junto com um parâmetro desejado para filtragem, como por exemplo, Tempo = 400 (min). Sempre que um campo de *Input* for informado, este será também analisado em relação ao Sucessor e sua

confiança mínima. Esta etapa possibilita melhor eficácia na execução do algoritmo e determina a análise de co-relação através da estrutura informada, fornecendo assim maior controle sobre as várias possibilidades de enfoque. As telas que na aplicação efetuam esta parametrização aparecem ao final do texto, no Apêndice A.

- **Etapa 2 - geração de *itemsets***: esta é a fase de pré-processamento proposta por Agrawal *et al*. [14]. Aqui o primeiro conjunto de dados é gerado pelo algoritmo. São definidos os conjuntos iniciais de relações baseados somente no suporte mínimo, e gerados os itens freqüentemente relacionados através dos parâmetros informados na estrutura. Para a carcaterização da informação como transação do tipo *basket*, primeiramente identifica-se a transação e o "item" da "cesta", através dos parâmetros Chave e Sucessor da regra, respectivamente, informados no cadastro da estrutura (Tabela 3.1.). A seguir é feita a seleção desta relação na tabela UsoSoftware, descartando-se os *itemsets* com ocorrência inferior ao suporte mínimo informado e atribuindo a cada *itemset* sua freqüencia diante do conjunto total. Este conjunto então é armazenado em tabela temporária de estrutura do tipo *basket,* possibilitando assim a correta execução do *Apriori* sobre os dados. Por exemplo, para um total de 50.000 transações:, a Tabela 3.2 mostra 3 itemsets gerados a partir da base de dados do MoonGrid:

Tabela **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.2: *Itemsets* gerados pelo MoonGrid M.Man – exemplo 1
Fonte: Autoria Própria

Suporte	Itemset
1643 (3,29%)	Programa (4,5,6) , Hora (5:30 pm)
1003 (2,01%)	Programa (1,2,3) , Tempo (entre 395 e 471 min)
360 (0,72%)	Programa (1,2) , Tempo (< 204 min) , Data (25/10/2007)

Considera-se esta etapa do processamento a principal responsável por um satisfatório desempenho do algoritmo *Apriori* sobre uma base de dados relacional tradicional, pois a maior dificulade de se implementar o Apriori diz respeito à como os dados estão organizados.

A tabela acima apresenta *Itemsets* relacionando programa utilizado e hora de execução, programa utilizado e tempo de utilização, e por fim programa utilizado, tempo de utilização e data da execução, de acordo com os parâmetros demonstrados pela Tabela 3.1. A coluna Suporte indica a freqüência de incidência da relação mostrada na segunda coluna. A identificação dos programas é apresentanda somente como número por questões de confidencialidade dos dados.

- **Etapa 3 - geração das regras de associação**: etapa responsável pela leitura final do conjunto de transações para que então seja feita a construção das regras tendo como critério principal a verificação do parâmetro “Sucessor da Regra”, informado na estrutura do modelo, diante da cadeia de *Itemsets* gerados. Para o exemplo 1 mostrado nas etapas anteriores, podemos verificar as regras obtidas através da Tabela 4.5 demonstrada no próximo capítulo, onde a geração de regras é melhor detalhada.

As duas principais etapas de processamento (Apêndice B) foram implementadas diretamente sobre a base de dados, em T-SQL, a linguagem de scripts do SQL Server. O processo de amostragem delimita subconjuntos de intervalos de tempo aleatórios.

4 DADOS EXPERIMENTAIS

Este capítulo apresenta os resultados de dois experimentos realizados com um subconjunto de dados extraídos da base do MoonGrid no Banrisul. O primeiro conjunto de experimentos foi útil para mostrar quais seriam os principais pontos críticos para a escalabilidade da aplicação, visando à relevância dos resultados gerados.

Todos os experimentos descritos aqui foram realizados em um PC Pentium dual-core de 1.73 Ghz e 1MB de memória RAM. Os dados utilizados e apresentados são frações de um conjunto total de 3.533.485 de registros para a tabela UsoSoftware.

4.1 Experimentos preliminares

O objetivo destes experimentos foi testar as primeiras versões do módulo proposto (anteriores à versão descrita no capítulo anterior, seção 3.3.1) e identificar os pontos críticos de desempenho para os testes futuros. Aqui foi medido o tempo computacional gasto em cada etapa do processo. As etapas envolvidas neste momento do desenvolvimento da aplicação, do ponto de vista técnico, poderiam ser descritas como apenas duas:

- **geração do primeiro conjunto**: aqui é feita a primeira leitura do conjunto de dados, selecionando os candidatos freqüentemente relacionados de acordo com o suporte mínimo.
- **relacionamento entre *itemsets***: nesta etapa, o processo analisa a relação de cada candidato em relação ao elemento sucessor da regra e ao parâmetro confiança.

O tempo computacional do processo está apresentado na Tabela 4.1. Nas duas últimas colunas é apresentado o tempo em minutos. Para esta medição, o suporte mínimo utilizado foi 0.2479 e somente o atributo Programa foi utilizado como sucessor da regra. O valor 0.2479 foi determinado como o menor melhor suporte em relação ao atributo Programa, do período abaixo avaliado.

Tabela **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.1: Tempo Computacional das Etapas do Processo

Mês/Ano	Qt. Transações	1ª Etapa min:seg	2ª Etapa min:seg	2ª Etapa – Média min:seg	2ª Etapa – D. Padrão
02/08	381931	00:04	10:36	10:33	0,0051
08/07	426733	00:04	11:30	11:39	0,0132
10/07	600342	00:06	16:30	16:45	0,0289

Fonte: Autoria Própria

A Tabela 4.1 mostra que a cada 50.000 transações o tempo de processamento da 2ª etapa tende a crescer aproximadamente 1 minuto. Este número representa atualmente a quantidade média de transações do período de dez dias no ambiente Banrisul. Portanto, na intenção de apoiar também um sistema escalonador de recursos considerou-se necessária alguma melhoria aplicada à versão inicial do algoritmo.

A partir dos resultados destes experimentos, iniciou-se a fase de melhoria no código do algoritmo e inclusão dos parâmetro adicionais para determinar maior precisão dos resultados. Este trabalho resultou na sub-divisão da primeira etapa com a implementação do que chamamos de pré-processamento.

4.2 Geração de regras

m segundo experimento, mais detalhado, foi realizado com o objetivo de demonstrar os resultados finais obtidos pelos esforços empregados ao longo do desenvolvimento deste trabalho. Os dados apresentados a seguir contemplam a versão final do módulo proposto e incluem basicamente as três etapas já detalhadas na seção 3.3.1 do capítulo anterior:

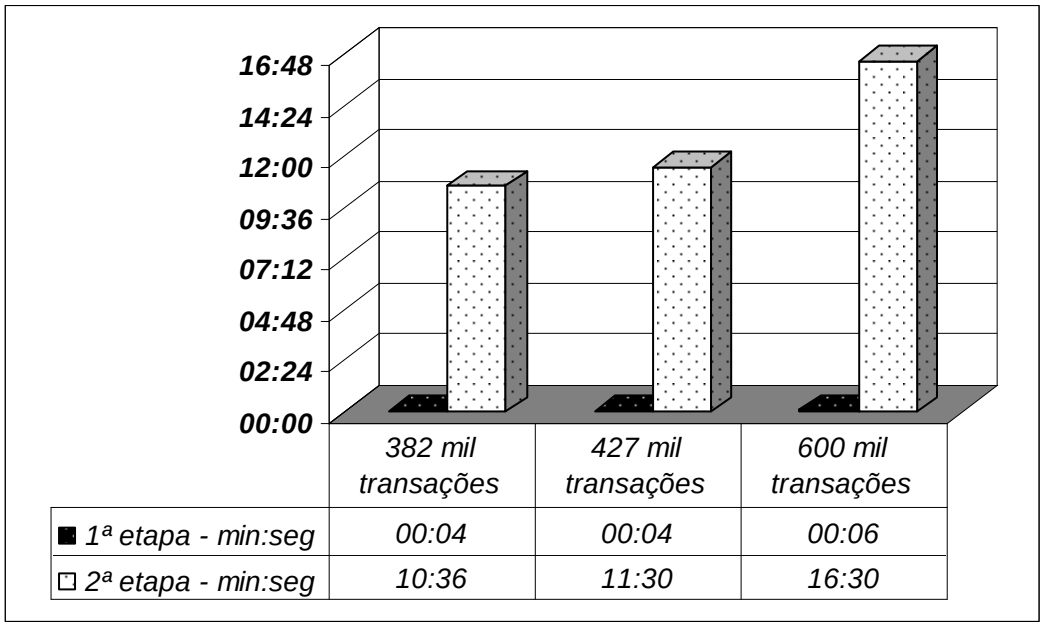
- **a fase de parametrização**, onde são indicadas as entidades relacionadas e atributos de Input para análise de freqüência sob a transação;
- **a fase de pré-processamento**, onde é feita a caracterização dos dados de forma transacional, atribuição de freqüência aos *itemsets* e descarte dos dados que não interessam ao restante do algoritmo;
- **a fase final**, que resulta na geração e apresentação das regras.

A Tabela 4.2, que serve de comparação à Tabela 4.1 exibida na seção anterior, mostra o tempo computacional gasto nas etapas do processo, com suporte mínimo igual a 0.2479 em relação ao atributo Programa. Não é demonstrado na tabela a etapa inicial do processamento, pois não sofreu nenhuma alteração.

Tabela **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.2: Tempo Computacional das Etapas do Processo após melhorias
Fonte: Autoria Própria

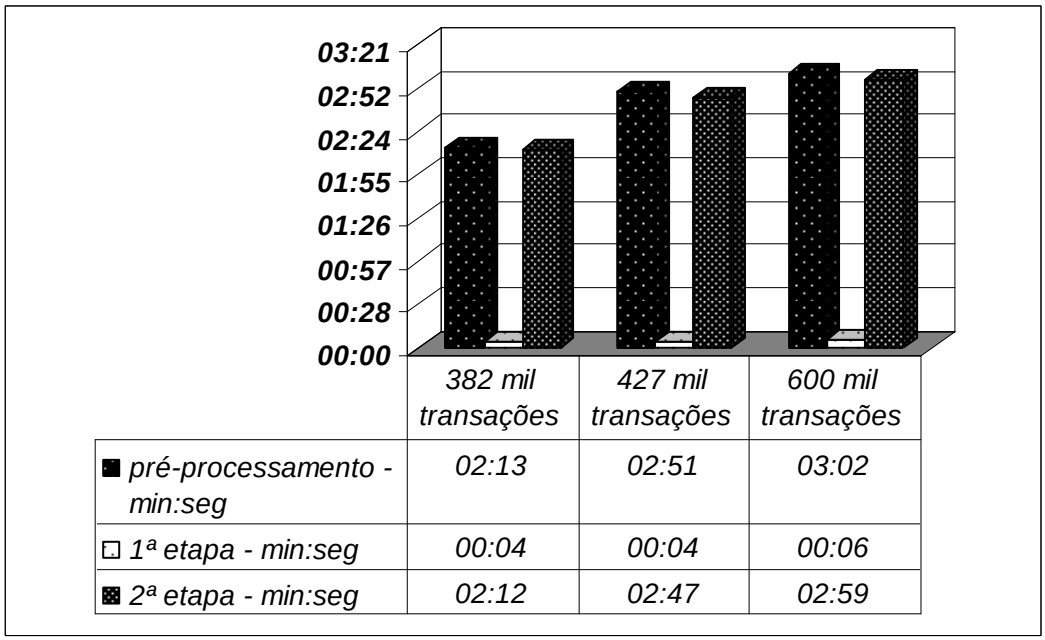
Mês/Ano	Qt. Transações	Pré-proc. min:seg	Pré-proc. Média min:seg	Pré-proc. D. Padrão	2ª Etapa min:seg	2ª Etapa Média min:seg	2ª Etapa D. Padrão
02/08	381931	02:13	02:18	0,0067	02:12	02:11	0,0020
08/07	426733	02:51	02:46	0,0062	02:47	02:47	0,0050
10/07	600342	03:02	03:04	0,0031	02:59	03:00	0,0018

Para ficar mais evidente a relação, as figuras 4.1 e 4.2 apresentam de forma visual os dados de ambas as tabelas:



	382 mil transações	427 mil transações	600 mil transações
1ª etapa - min:seg	00:04	00:04	00:06
2ª etapa - min:seg	10:36	11:30	16:30

Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**..1: Gráfico relativo à Tabela 4.1
Fonte: autoria própria.



	382 mil transações	427 mil transações	600 mil transações
pré-processamento - min:seg	02:13	02:51	03:02
1ª etapa - min:seg	00:04	00:04	00:06
2ª etapa - min:seg	02:12	02:47	02:59

Figura **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**..2: Gráfico relativo à Tabela 4.2
Fonte: autoria própria.

Estes dados evidenciam uma melhoria considerável na execução do algoritmo *Apriori* após incrementos adicionados à sua versão clássica, principalmente na obtenção de dados transacionais e no descarte de registros não relevantes ao restante do processamento.

A seguir são apresentadas algumas regras obtidas no processo de geração, tendo como sucessor os atributos Programa e Tempo, suporte mínimo = 7, para um total aproximado de 50.000 transações:

Tabela **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.3: Regras obtidas em relação ao atributo Programa
Fonte: Autoria Própria

Probabilidade	Regra
0,889	Usuário (centralizadora), Tempo(> 547) → Programa (1,2)
0,568	Usuário (cambio), Hora(5:27 pm) → Programa (4)
0,558	Usuário (oper2), Tempo(> 394) → Programa(1,2)
0,430	Data (15/09/2007), Tempo (< 204) → Programa (4,5,6)

Os dados da tabela 4.3 mostram que o Usuário Centralizadora utiliza os Programas 1 e 2 por um tempo maior que 547 min. em 89% dos casos, o Usuário Cambio utiliza o Programa 4 as 5:27 da tarde em 57% dos casos, o Usuário Oper2 utiliza os Programas 1 e 2 por um tempo maior que 394 min. em 56% dos casos e em 15/09/2007 os Programas 4, 5 e 6 foram utilizados por um tempo maior que 204 min. em 43% dos casos.

Tabela **Erro! Nenhum texto com o estilo especificado foi encontrado no documento.**.4: Regras obtidas em relação ao atributo Tempo
Fonte: Autoria Própria

Probabilidade	Regra
1,000	Data (1/12/2007), Programa (4,6) → Tempo (< 190)
0,667	Usuário (sureg), Programa (6) → Tempo (< 216)
0,567	Usuário (oper2), Hora (08:19) → Tempo (< 403)
0,545	Usuário (sureg), Programa (1,2,3) → Tempo (> 520)

A Tabela 4.4 mostra que em 1/12/2007 os Programas 4 e 6 foram utilizados por um Tempo menor que 190 min. em 100% dos casos, o Usuário Sureg utiliza o Programa 6 por um Tempo menor que 216 min. em 67% dos casos, o Usuário Oper2 acessa o primeiro Programa as 8:19 da manhã e fica por um tempo menor que 403 min em 57% dos casos e o Usuário Sureg acessa os Programas 1, 2 e 3 por um tempo maior que 520 min. em 55% dos casos.

Como meio de verificar a corretude da informação gerada, consultas SQL foram realizadas diretamente sobre a base, de modo que se conseguisse obter a quantidade de ocorrências específicas. Como por exemplo:

```
SELECT COUNT(*) AS Quant
FROM   dbo.UsoSoftware INNER JOIN dbo.Computadores
   ON  dbo.UsoSoftware.patrimonio = dbo.Computadores.patrimonio
WHERE (dbo.UsoSoftware.idPrograma = 4 or dbo.UsoSoftware.idPrograma = 2)
       AND (dbo.Computadores.usuario = 'centralizadora')
```

Cabe ressaltar que várias regras representam associações pouco interessantes, e também que a geração das regras não representa por si só a solução para determinada questão. No entanto, a implementação do *Apriori* no ambiente MoonGrid mostrou-se eficaz e capaz de fornecer informação de alto nível, possibilitando uma análise detalhada sobre a relação de dependência e probabilidade de ocorrências de uso de software na Grade.

5 CONCLUSÃO

Neste trabalho foram descritas as principais características de um sistema de monitoramento de recursos em ambiente de grade, no contexto do projeto MoonGrid. Indo ao encontro das necessidades e sugestões para trabalhos futuros, foi apresentado um novo módulo que implementa *Data Mining* como forma de obter conhecimento útil. Apesar de construído sobre a arquitetura do MoonGrid, o Módulo de Análise pode se conectar a qualquer aplicação implementada no SQL Server.

O módulo foi testado utilizando a base histórica de acessos coletados a partir de Julho de 2007, no Banrisul. Com a evolução dos testes, modificações foram feitas para melhorar a usabilidade e desempenho da aplicação. Conforme os resultados obtidos pode-se dizer que a eficácia do algoritmo implementado é reconhecida como muito boa para os dados coletados pelo MoonGrid, porém a evidência de que as regras geradas são realmente interessantes para o ambiente depende de análise diante do que se deseja / necessita saber. Esta análise encontra-se fora do escopo deste trabalho.

Uma das tentativas iniciais para criação de um ambiente de monitoramento e testes foi utilizar o Laboratório 24h do La Salle, criando uma Grade com estrutura MoonGrid e sensores distribuidos. Porém, dificuldades como limitações do software SQL-Server Express 2005 para comunicação distribuída e limitações do próprio MoonGrid, pelo fato de ser implementado somente para sistema operacional Windows, impediram que se continuasse com esta estratégia. Portanto, ainda continua em aberto uma importante questão para continuidade do MoonGrid como tema e objeto de pesquisa: a portabilidade do ambiente para um sistema operacional e em sistema de banco de dados de utilização livre e *OpenSource*.

A criação do Módulo de Análise abriu algumas possibilidades que podem ser aprofundadas em trabalhos futuros, como por exemplo a implementação de outras

técnicas de *Data Mining* no ambiente do MoonGrid ou a definição de outro tipo de serviço, como utilizar determinadas regras ou informações para escalonamento de tarefas.

REFERÊNCIAS

HOLLINGSWORTH, J.; Tierney .B. Instrumentation and monitoring: In: Foster I, Kesselman C, editors. **The Grid:** blueprint of a new computing infrastructure. Second Edition. San Francisco: Morgan Kauffman; 2004.

LEMOS, D. T., **Monitoramento de recursos em ambientes de grade.** Trabalho de Conclusão de Curso;.Canoas: Ciência da Computação/UNILASALLE, 2006.

VARGAS, P. K., GRAND: **Um modelo de gerenciamento hierárquico de aplicações em ambiente de computação em grade.** Tese de Doutorado;. Rio de Janeiro: COPPE-Sistemas/UFRJ, 2006.

Open Grid Forum. Disponível em: <http://www.ogf.org/>.

Open Grid Forum. Performance Working Group. Final Document Number GFD.7. Disponível em: <http://www.ogf.org/documents/GFD.7.pdf>.

BOSCARIOLI ,C. **Análise de logs da web por meio de técnicas de data mining.** Paraná: UNIOESTE, 2003.

KENNY, S. et al. The CanonicalProducer: An instrument monitoring component of the Relational Grid Monitoring Architecture (R-GMA): **Sci. Program**. 13, 2 (Apr. 2005), 151-158.

The Globus Toolkit v4.2.0, The Globus Project. Disponível em: <http://www.globus.org/toolkit>.

FREITAS, A .A.; LAVINGTON, S. H. **Mining very large databases with parallel processing.** Kluwer Academic Publishers, 1998.

FELDMAN, R.; DAGAN, I. Knowledge Discovery in Textual Databases (KDT): In **Proc. of the 1st International Conference on Knowledge Discovery and Data Mining (KDD-95)**, 112 – 117. AAAI / MIT Press: Menlo Park, CA, 1995.

MATTOS, É. C. T. **Análise e construção de um ambiente de grade computacional peer-to-peer com ênfase no balanceamento de carga:** Tese de doutorado, Universidade Federal de São Carlos, São Carlos, 2003.

KRAUTER, K.; BUYYA, R.; MAHESWARAN, M. A taxonomy and survey of grid resource management systems for distributed computing: **Software – Practice and Experience**, v. 32, n. 2, pp. 135–164, 2002.

AGRAWAL R.; SRIKANT R. Fast algorithms for mining association rules: **VLDB.** Sep 12-15 1994, Chile, 487-99, ISBN 1-55860-153-8.

AGRAWAL, R.; IMIELINSKY, T.; SWAMI, A. **Mining association rules between sets of items in large databases:** proc. of the int. conf on management of data (SIGMOD-93), 207-216. Washington, DC, USA, 1993.

INMON, W. H.; WELCH, J. D.; GLASSEY, K. L. **Gerenciando data warehouse:** técnicas práticas para monitorar operações de performance, administrar dados e ferramentas, gerenciar alterações e crescimento. São Paulo: Makron Books, 1999.

DAMASIO, F. W. **Um estudo de interatividade em sistemas operacionais e sua aplicação no kernel Linux:** São Leopoldo, RS, UNISINOS, 2005.

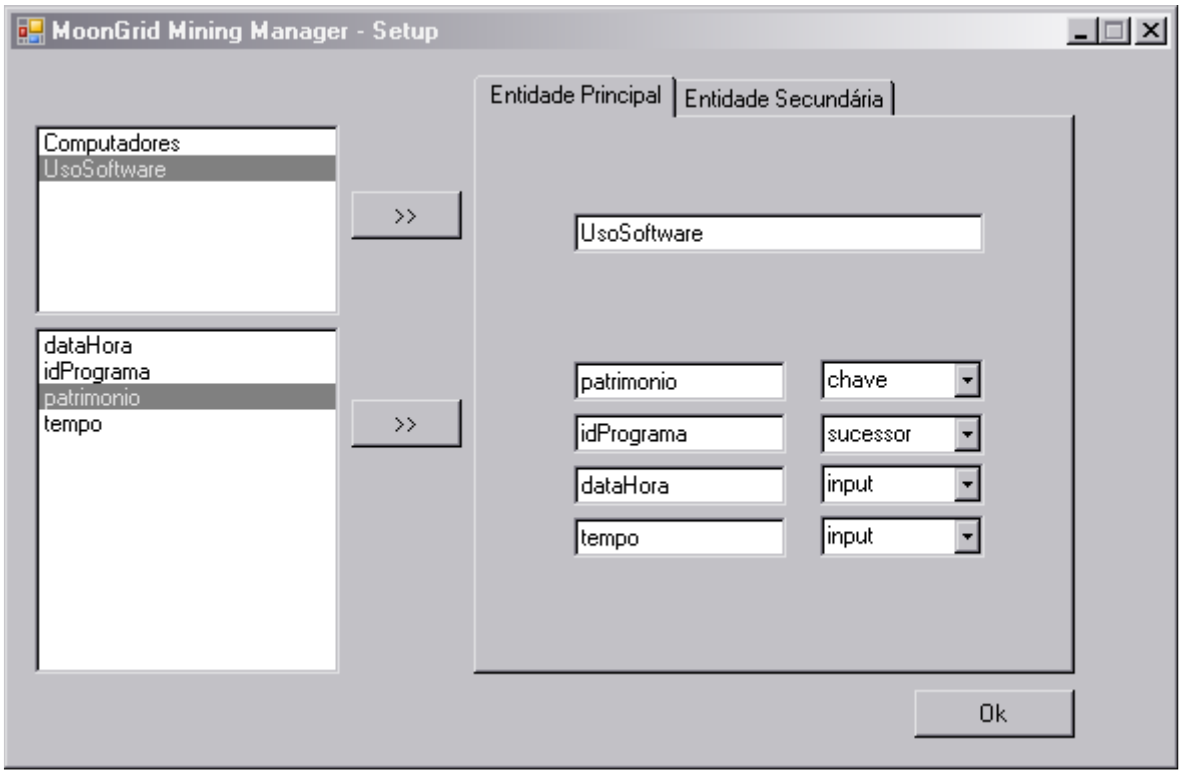
WITTEN, I. H. **Data mining:** practical machine learning tools and techniques: 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

GONÇALVEZ, E. C. **Regras de associação e suas medidas de interesse objetivas e subjetivas:** Niterói, RJ, UFF, 2006.

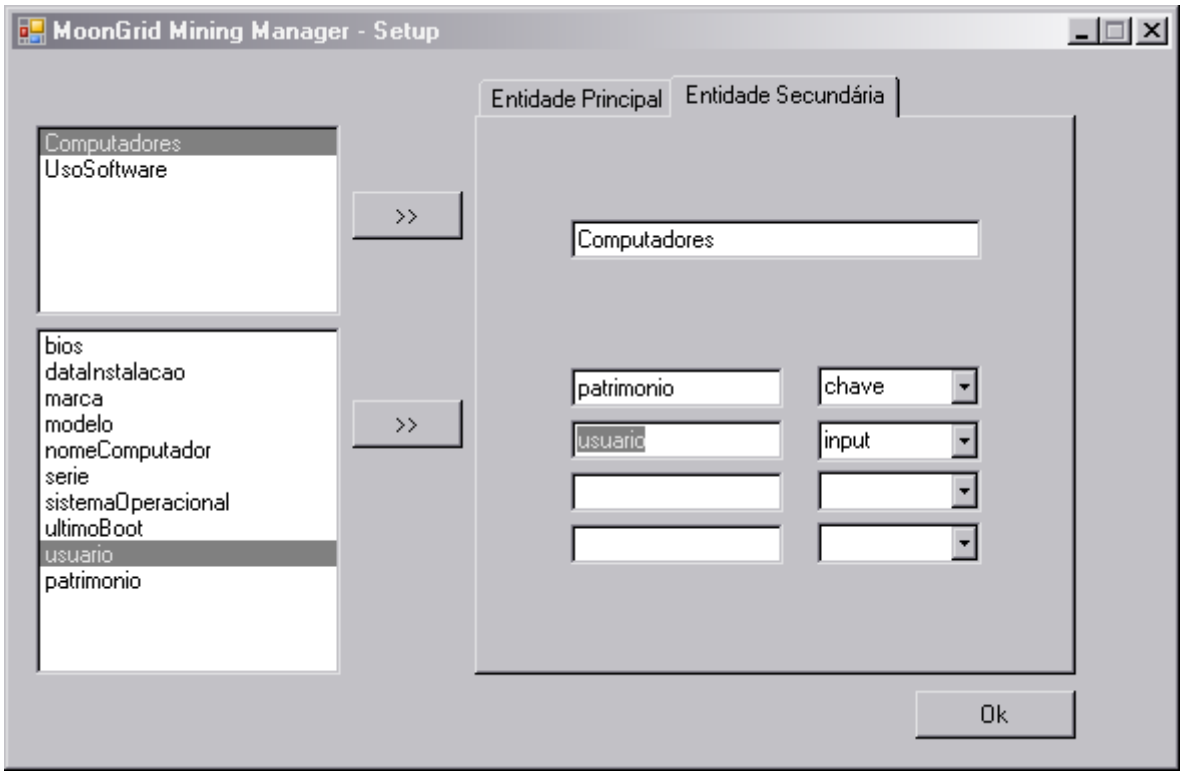
Condor Project, (2006a). *Hawkey*. Disponível em: <http://www.cs.wise.edu/condor/hawkey/>.

Ganglia, (2006). **Ganglia monitoring system.** Disponível em: <http://www.ganglia.sourceforge.net/>.

APÊNDICE A – TELAS DE CONFIGURAÇÃO DO MÓDULO



Configuração de Entidade Principal



Configuração de Entidade Secundária

APÊNDICE B – ROTINAS PRINCIPAIS DO ALGORITMO APRIORI

```
--------------------------------------
-- código fonte dos scripts T-SQL
-- rotinas principais
--------------------------------------


set ANSI_NULLS ON
set QUOTED_IDENTIFIER ON
go

ALTER PROCEDURE [dbo].[F_APRIORI] @SUP_MIN NUMERIC(10,2)
AS
BEGIN
	SET NOCOUNT ON;

	DECLARE @K INT
	DECLARE @N BIGINT

	CREATE TABLE #TMP_GERAL
	(
		ITENSET VARCHAR(),
		FREQ NUMERIC(10,2)
	)

	-- determinando os itens frequentes
	-- atribuindo a cada um dos itens seu suporte

	SELECT CAST(dbo.InfoTable.idSucessor AS VARCHAR(10)) AS ITENSET,
		 COUNT(*)/CONVERT(NUMERIC(10,2), @N) AS FREQ
	INTO #TMP
	FROM @dbo.InfoTable
	GROUP BY @dbo.InfoTable.idSucessor
	HAVING ( COUNT(*)/CONVERT(NUMERIC(10,2),@N) ) >= @SUP_MIN

	INSERT #TMP_GERAL
	SELECT * FROM #TMP


	-- calculados os subgrupos com suporte maior que o suporte informado
	-- os grupos finais são armazenados na #tmp_geral

	SET @K=2

	WHILE EXISTS(SELECT * FROM #TMP)
	BEGIN

		-- gera subgrupos com K elementos
		EXEC F_APRIORI_GEN @K

		-- retorna a frequencia
		EXEC F_FREQUENCIA @K, @N

		-- descarta os itemsets que não interessam
		DELETE #TMP
		WHERE FREQ < @SUP_MIN

		-- adiciona aqui os itens restantes
		-- que interessam ao suporte informado

		INSERT #TMP_GERAL
		SELECT * FROM #TMP

		SET @K = @K + 1
	END

	SELECT * FROM #TMP_GERAL

END
```

```
-------------------------------------
-- rotina para obter o k-itemset
-- e retornar os itemsets candidatos
-------------------------------------

ALTER PROCEDURE [dbo].[F_APRIORI_GEN] @QTD INTEGER
AS
BEGIN
	SET NOCOUNT ON;

	DECLARE @I VARCHAR()
	DECLARE @J VARCHAR()

	CREATE TABLE #TMP_RET
	(
		COL1 VARCHAR()
	)


	SELECT DISTINCT ITENSET
	INTO #H
	FROM #TMP

	DECLARE C_I CURSOR FOR
	SELECT ITENSET
	FROM #TMP

	OPEN C_I

	FETCH NEXT FROM C_I
	INTO @I

	WHILE @@FETCH_STATUS = 0
	BEGIN


		-- selecionar somente o
		-- itemset maior que o último elemento em I

		DECLARE C_J CURSOR FOR
		SELECT ITENSET
		FROM #H
		WHERE ITENSET > SUBSTRING(@I,dbo.RAT(',',@I,1) + 1,LEN(@I) )

		OPEN C_J

		FETCH NEXT FROM C_J
		INTO @J

		WHILE @@FETCH_STATUS = 0
		BEGIN

			INSERT #TMP_RET VALUES(@I + ',' + @J)

			FETCH NEXT FROM C_J
			INTO @J
		END

		CLOSE C_J
		DEALLOCATE C_J


		FETCH NEXT FROM C_I
		INTO @I
	END

	CLOSE C_I
	DEALLOCATE C_I

	TRUNCATE TABLE #TMP

	INSERT #TMP
	SELECT *
	FROM  #TMP_RET
END
```

```
-------------------------------------
-- calcula a frequencia dos itemsets
-- gerados anteriormente
-------------------------------------

ALTER PROCEDURE [dbo].[ST_FREQUENCIA] @QTD INT, @N INT
AS
BEGIN
	SET NOCOUNT ON;

	DECLARE @I VARCHAR()
	DECLARE @J VARCHAR()

	DECLARE @F NUMERIC(10,2)
	DECLARE @CNT BIGINT

	CREATE TABLE #TMP_TOTAL
	(
		ITENS VARCHAR()
	)

	CREATE TABLE #TMP_ITEMSET
	(
		ITENS VARCHAR()
	)


	DECLARE C_I CURSOR FOR
	SELECT ITENSET
	FROM #TMP

	OPEN C_I

	FETCH NEXT FROM C_I
	INTO @I

	WHILE @@FETCH_STATUS = 0
	BEGIN

		TRUNCATE TABLE  #TMP_ITEMSET

		INSERT  #TMP_ITEMSET
		SELECT DISTINCT str AS ITENSET
		FROM dbo.iter_charlist_to_tbl(@I,',' )

		DECLARE C_J CURSOR FOR
		SELECT CAST(idSucessor as varchar(2))
		FROM dbo.InfoTable

		OPEN C_J

		FETCH NEXT FROM C_J
		INTO @J

		SET @CNT = 0

		WHILE @@FETCH_STATUS = 0
		BEGIN

			TRUNCATE TABLE #TMP_TOTAL

			INSERT #TMP_TOTAL
			SELECT DISTINCT str AS ITENSET
			FROM dbo.iter_charlist_to_tbl(@J,',' )

			IF (SELECT COUNT(*)
			    FROM #TMP_TOTAL AS A, #TMP_ITEMSET AS B
			    WHERE A.ITENS = B.ITENS ) = @QTD
			BEGIN
				SET @CNT = @CNT + 1
			END

			FETCH NEXT FROM C_J
			INTO @J
		END

		CLOSE C_J
```

```
                DEALLOCATE C_J

                SET @F =@CNT / CONVERT(NUMERIC(10,2), @N)

                UPDATE #TMP
                SET FREQ = @F
                WHERE ITENSET = @I

                FETCH NEXT FROM C_I
                INTO @I
        END

        CLOSE C_I
        DEALLOCATE C_I

END
```