Classificação Personalizada de Nós de Grid Utilizando Redes Neurais Artificiais

Dniester Amorim de Aguiar
Universidade de Pernambuco
Recife, Brasil
daa2@ecomp.poli.br

Byron Leite Dantas Bezerra
Universidade de Pernambuco
Recife, Brasil
byronleite@ecomp.poli.br

Prof. Sérgio Galdino
Universidade de Pernambuco
Recife, Brasil
sergio.galdino@ieee.com

***Resumo*—O uso de Sistemas de Computação Distribuída tem aumentado recentemente. *Grids* e *clusters* se tornaram as melhores opções para a maior parte das necessidades de computação de alto desempenho devido ao seu baixo custo e fácil implementação. Hoje, grids e clusters são utilizados em várias áreas de pesquisa, tais como medicina, química e física, e muitas destas fazem uso de grandes grids colaborativos, como o *Worldwide LHC Computing Grid*. Dada a grande variedade de hardware na composição dos *grids*, uma classificação personalizada dos nós facilitaria o gerenciamento, porém as constantes mudanças na estrutura do *grid* torna a classificação manual praticamente impossível. Este paper apresenta uma forma de classificar os nós automaticamente baseado em regras personalizadas utilizando uma rede neural para aprender as regras de classificação desejadas e classificar os nós que se unem ao *grid*.**

Palavras-chave: Grid Computing, Classificação, Redes Neurais

I. Introdução

Com a crescente evolução das tecnologias de rede, é cada vez maior o interesse nas soluções de alto desempenho baseadas em computação distribuída[1]. Dentre elas, as mais populares são os *grids* e os *clusters*, que permitem a implementação de um sistema de computação de alto desempenho através do uso de paralelismo, tornando possível obter um alto poder de processamento com pouca dificuldade e custo significantemente baixo quando comparados aos de um supercomputador, tornando possíveis grandes avanços nas mais diversas áreas da ciência.

No entanto, o gerenciamento e o uso eficiente desses sistemas podem ser bastante árduos, principalmente em *grids* de grande tamanho, como os vários *grids* colaborativos existentes, onde os nós são diferentes uns dos outros. Dependendo do caso, pode acabar havendo desperdícios de recursos, pois por não haver uma classificação adequada, nós com um grande poder computacional podem acabar sendo usado para tarefas menos importantes, quando um determinado procedimento poderia se beneficiar muito mais do *hardware* mais potente. Neste exemplo, classificar e agrupar os nós de acordo com seu poder computacional possibilitaria delegar tarefas que exigissem um tempo de resposta menor para nós de maior desempenho. Em outro exemplo, nós que possuíssem adaptadores de vídeo potentes, capazes de acelerar o processamento de imagens poderiam ser agrupados de forma a serem utilizados prioritariamente para tais tarefas, evitando que fossem ocupados com outros processamentos.

No entanto, uma classificação manual de nós em um *grid* de grande tamanho é impraticável. Além da quantidade de nós, a natureza dinâmica desses sistemas tornaria o trabalho interminável. Enquanto um sistema de classificação automática tradicional possa ser capaz de resolver em casos mais simples, regras de classificação mais complexas demandariam um custo de tempo e afetariam o desempenho do sistema.

Por isso, neste trabalho, propomos um sistema de classificação automática utilizando uma rede neural artificial, de forma a prover uma solução genérica e simples, capaz de suportar regras de classificação complexas sem consumir muitos recursos.

Este artigo está organizado de acordo com as seções a seguir. Uma breve introdução sobre *Grid Computing* é apresentada na Seção II. Na Seção III é apresentada uma visão geral sobre o uso de redes neurais em problemas de classificação. Na Seção IV mostramos alguns trabalhos relacionados. A Seção V apresenta o modelo proposto. Em seguida, os resultados dos experimentos são detalhados na Seção VI. Por fim, apresentamos nossas conclusões na Seção VII.

II. *Grid Computing*

Um sistema de computação em grade (*Grid Computing System*) é um ambiente distribuído geograficamente com domínios autônomos que compartilham recursos entre si [2]. Em linhas gerais, se trata de uma tecnologia que permite o compartilhamento em larga escala de recursos distribuídos geograficamente, de forma a possibilitar a resolução de problemas que necessitem de alto poder computacional.

Num ambiente de *grid*, recursos distribuídos são descobertos, descritos, selecionados, alocados, compartilhados e agregados para grandes tarefas computacionais, como experimentos de física energética, pesquisas astronômicas, processamento digital de imagens e vídeo, etc. Estes recursos podem ser recursos computacionais, dispositivos de armazenamento ou sensores. Um *grid* também pode ser considerado como um conjunto de *clusters* posicionados em diferentes localidades geográficas, onde cada *cluster* comporta um conjunto de recursos.

Um usuário desenvolve uma aplicação para o *grid* de acordo com sua necessidade, e essa aplicação é composta por uma determinada quantidade de tarefas. Uma tarefa pode ser submetida de qualquer nó no *grid*. O sistema de gerenciamento do *grid* se encarrega de lidar com as diferenças de hardware, de

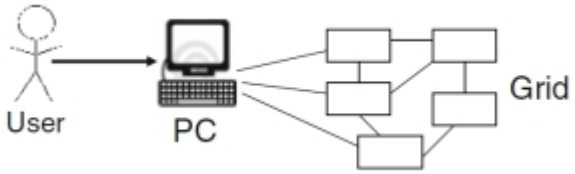

Figura 1. Esquema simplificado de um *grid*

software e de protocolos de comunicação, provendo serviços para funções básicas.

Grids são classificados em muitas formas. As duas principais categorias de *grids* são os *Grids* Computacionais e os *Grids* de Dados. Os *grids* Computacionais são os sistemas que possuem uma maior capacidade computacional agregada para aplicações únicas do que qualquer unidade isolada do sistema. Já os *Grids* de Dados, é para sistemas que proveem uma infraestrutura para sintetizar novas informações a partir de repositórios de dados disponíveis.

Em linhas gerais, podemos dividir um sistema de *grid computing* em seis componentes:

- Interface de Usuário: Elemento responsável por permitir ao usuário iniciar aplicações que irão fazer uso dos recursos e serviços oferecidos pelo *grid*.
- Segurança: Um requisito de grande importância para um ambiente de *grid*. Mecanismos de segurança, incluindo autenticação, autorização, criptografia entre outros devem ser implementados, tanto na iteração com o usuário como na comunicação entre os nós.
- *Broker*: Uma vez autenticado, o usuário irá iniciar uma aplicação. Baseado na aplicação iniciada e e outros parâmetros dados pelo usuário, o próximo passo é identificar os recursos apropriados disponíveis para serem utilizados. Esta tarefa é executada pelo *broker*.
- Agendador (*Scheduler*): Uma vez que os recursos tenham sido identificados, o próximo passo lógico é o agendamento das tarefas associadas à aplicação para serem executadas. Se o conjunto de tarefas não possuem interdependência entre si, então um agendador simples poderá ser suficiente. caso contrário, um agendador de processos mais complexo deverá ser empregado.
- Gerenciamento de Dados: Se qualquer dado, incluindo módulos de aplicação devem ser movidos ou devem se tornar acessíveis aos nós onde a aplicação será executada, então será necessário prover um método seguro e confiável para transmitir os dados.
- Gerenciamento de Tarefas e Recursos: O principal componente de um sistema *grid*. É responsável pro prover os serviços para iniciar uma tarefa em um recursos, verificar o estado e receber os resultados quando a mesma for completada.

Segundo Chopra [3], as características principais de um ambiente de *grid* são:

- Heterogeneidade: Agrega grandes quantidades de recursos computacionais e de informação independentes e distribuídos geograficamente.
- Dinamismo: O ambiente de computação, informação e comunicação em *grid* estão em constante mudança durante a vida de uma aplicação.
- Incerteza: Dinamismo, falhas e conhecimento incompleto do estado global levam a incerteza em um ambiente de *Grid*
- Segurança: Segurança (autenticação, autorização e controle de acesso) e confiança são desafios críticos em ambientes de *grid*

III. Classificadores MLP

Recentemente, redes neurais tem sido usadas em várias aplicações, tais como classificação [4], previsão [5] e aplicações biomédicas [6]. A popularidade das redes neurais se dá em parte devido a sua habilidade de aprender e generalizar. Em particular, redes neurais não fazem nenhuma dedução sobre a estatística dos dados de entrada, e são capazes de realizar tomadas de decisão complexas [7]. Essa propriedade faz as redes neurais serem uma solução atrativa para muitos problemas de classificação.

Em redes neurais, a escolha do algoritmo de aprendizagem, da topologia da rede, da inicialização dos pesos e da representação dos sinais de entrada são fatores importantes para o desempenho do aprendizado. Em particular, a escolha do algoritmo de aprendizado determina a taxa de convergência e o custo computacional da solução. Dentre as estruturas de rede, as redes MLP (*Multilayer Perceptron*) estão entre as mais usadas.

A MLP é uma estrutura de rede sem realimentação (*Feedforward*) onde os elementos são conectados apenas entre as duas camadas adjacentes. Os elementos da camada oculta são caracterizados por uma função de ativação sigmóide:

$$y_i = f(net_i) \tag{1}$$

O sinal net_i é a soma ponderada dos sinais de entrada do i-ésimo elemento, $net_i = \sum_j w_{ij} y_j$, onde w_{ij} é o peso associado à ligação entre os elementos j e i. O aprendizado consiste na minimização do erro euclidiano para todos os pares entrada-saída do conjunto de treino.

$$E(w) = 0.5 \sum_{k=1}^{p} \left\| y^{(k)}(w) - d^{(k)} \right\|^2 \tag{2}$$

onde y é o vetor de saída da rede neural e d o vetor de saída esperado. O vetor w representa os pesos ajustados e p é a quantidade de pares entrada-saída. Os pesos podem ser treinados na fase de aprendizado da rede usando qualquer algoritmo de retropropagação (*backpropagation*). Após o treinamento, os pesos são fixados, e a rede estará pronta para uso.

IV. Trabalhos Relacionados

O gerenciamento de *grids* computacionais tem sido tema de pesquisas recentes, buscando melhorar ou facilitar a manutenção e a administração destas plataformas. Dentre eles, destacamos alguns trabalhos relacionados.

Em [8], os autores propuseram um sistema de monitoração de tarefas, capaz de fornecer informações sobre o estado da

tarefa submetida, bem como dados sobre o nó ao qual foi associado a tarefa, durante a execução da mesma, facilitando a prevenção, controle e correção de erros e consequentemente, ajudando a melhorar a estabilidade do sistema. A principal contribuição do trabalho foi a criação do sistema proposto, incluindo uma interface gráfica integrada com o sistema de monitoração de tarefas, permitindo que os administradores do grid pudessem acompanhar a execução das tarefas em tempo real, bem como identificar qualquer erro que ocorra durante o processo.

Em [9], Yang, Galis e Todd propuseram introduzir o gerenciamento baseado em políticas, aprimorada através do uso de uma rede ativa para prover um mecanismo de gerenciamento que permita o *grid* se adaptar a vários requisitos de aplicação de forma flexível e automática, reduzindo o esforço de execução das tarefas. Com isto, os autores conseguiram facilitar a administração de um grid, reduzindo a necessidade de intervenção humana durante a execução das tarefas, através do uso de políticas de auto-escalamento.

Carstoiu e Cerninan [10] propuseram um *framework* para gerenciar *grids* voluntários, que tenta definir mecanismos que asseguram funcionalidade correta para sistemas baseados no conceito de *Grids Voluntários*. Com isso, a abordagem ajuda a resolver problemas específicos desta classe de *grids*, tais como tentativas de sabotagem originadas dos nós. O trabalho contribui de forma significante com a administração de *grids* voluntários, no entanto, outros tipos de *grids* computacionais podem não se beneficiar tanto das técnicas apresentadas.

Hao Tian [12] Sugeriu um sistema de gerenciamento de tarefas e recursos para *grids* computacionais utilizando um algoritmo híbrido das técnicas de algoritmos genéticos e de colônia de formigas como uma alternativa eficiente para alcançar um melhor desempenho na alocação das tarefas aos nós comparado aos métodos tradicionais. O modelo proposto apresentou boa expansibilidade e bom desempenho, principalmente em grids com quantidades grandes de tarefas e nós.

V. Descrição do Modelo

Quando um novo nó é adicionado ao *grid*, o sistema de gerenciamento de recursos obtém informações sobre os nós, que é usada para selecionar um nó que possua recursos suficientes para executar uma determinada tarefa. Esses dados incluem o *clock* e número de núcleos dos processadores, a quantidade de memória RAM instalada e o espaço em disco disponível, dentre outras informações. Nossa proposta utiliza-se dessas informações obtidas para classificar e agrupar os nós em classes definidas pelo administrador do sistema.

No entanto, classificar os nós em *grids* muito grandes ou muito dinâmicos, como os *grids* colaborativos, é uma tarefa impossível de se realizar manualmente. E devido aos dados e regras de classificação utilizadas serem diferentes dependendo do ambiente, uma solução de classificação tradicional seria inviável, devido ao alto custo de processamento. Para solucionar estes problemas, nossa abordagem utiliza-se de uma rede neural artificial.

Em um primeiro momento, a rede neural usa um conjunto de informações pré-definidas para aprender as regras de classificação. A composição deste conjunto deve ser definido apropriadamente, de acordo com as regras desejadas e com os dados a serem utilizados. O tamanho do conjunto também é importante, pois como o administrador deverá classificar manualmente este conjunto, ele não deverá ser muito grande, pois inviabilizaria a abordagem, nem muito pequeno, para que haja dados suficientes para que o sistema aprenda as regras de classificação adequadamente. Após a classificação manual das informações, o conjunto é utilizado para treinar a rede neural com um método supervisionado, de forma que ela aprenda as regras de classificação desejadas.

A estrutura da rede neural deverá ser definida de forma a ser capaz de aprender as regras de classificação e de processar os dados de entrada. Sendo assim, a camada de entrada deverá possuir um elemento para cada dado de entrada. De maneira análoga, a camada de saída deverá possuir elementos suficientes para permitir que a rede possa prover as saídas desejadas. Para a camada escondida, no entanto, a quantidade deverá ser encontrada em grande parte por um método de tentativa-e-erro, porém algumas regras podem ser utilizadas para ajudar a encontrar a composição correta, como descrito em [11].

VI. Resultados

Nos experimentos, o objetivo foi classificar nós de um *grid* hipotético com 29 nós com diferentes especificações de hardware. Nós utilizamos o *clock* efetivo do processador (calculado através da multiplicação do *clock* base pela quantidade de núcleos do processador), a quantidade de memória RAM,

Tabela I. Composição da Base Usada nos Experimentos

Tipo de Nó	**Quantidade de Nós**
Nós de Alto Desempenho:	17
Nós de Médio Desempenho:	19
Nós de Baixo Desempenho:	20
Total de Nós:	56

Tabela II. Composição do Conjunto de Treino

Tipo de Nó	**Quantidade de Nós**
Nós de Alto Desempenho:	8
Nós de Médio Desempenho:	8
Nós de Baixo Desempenho:	11
Total de Nós:	27

Tabela III. Composição do Conjunto de Avaliação

Tipo de Nó	**Quantidade de Nós**
Nós de Alto Desempenho:	9
Nós de Médio Desempenho:	11
Nós de Baixo Desempenho:	9
Total Nodes:	29

Tabela IV. Resultado dos Experimentos

Classificações Corretas	742
Classificações totais	870
Precisão Média	0.8528736
Desvio Padrão	0.06269556
t-value	74.509
p-value	2.2e-16
Intervalo de Confiança (95%)	[0.8294627,0.8762845]

e o *benchmark* do adaptador gráfico para classificar os nós em três classes:

1) Nós de alto desempenho
2) Nós de médio desempenho
3) Nós de baixo desempenho

Para encontrar a quantidade apropriada de elementos na camada escondida, nós utilizamos as regras sugeridas em [11]:

1) A quantidade de elementos na camada escondida deverá estar entre a quantidade de elementos na camada de entrada e a quantidade de elementos na camada de saída
2) A quantidade de elementos na camada escondida deverá ser $\frac{2}{3}$ da quantidade de elementos na camada de entrada, somada à quantidade de elementos na camada de saída
3) A quantidade de elementos na camada escondida deverá ser menor que o dobro da quantidade de elementos na camada de entrada.

Sendo, assim a estrutura da rede utilizada foi:

- Três elementos na camada de entrada
- Quatro elementos na camada escondida
- Dois elementos na camada de saída

A técnica utilizada para treinar a rede neural foi a de Retropropagação Resiliente (*Resilient Backpropagation*). Nós escolhemos como condição de parada do algorítmo a taxa de erro de treino ser menor que 5%, pois em nossos experimentos, taxas menores resultaram em super treinamento (*overfitting*).

Para os experimentos, nós usamos uma base particular de especificaçoes composta de 56 nós, descritos na Tabela I. Essa base foi usada para formar o conjunto predefinido de treino e o conjunto de avaliação. Suas composições estão descritas nas tabelas II e III, respectivamente.

O experimento foi executado 30 vezes. Em cada execução, nós randomizamos os pesos, treinamos a rede com o conjunto predefinido, executamos a rede no modo *feedforward* utilizando o conjunto de avaliação, e registramos a taxa de acerto. A rede apresentou uma precisão aceitável, dado o pequeno conjunto de exemplos usado no treino, como mostra a Tabela IV.

VII. Conclusão e Trabalhos Futuros

Os experimentos demonstraram que é possível usar uma rede neural para classificar os nós de um *grid*, utilizando regras definidas pelo administrador, com uma aceitável precisão, mesmo com um conjunto de treino razoavelmente pequeno. Em casos onde a taxa de erro precise ser menor, o aumento do conjunto de treino, ou um melhor refinamento dos seus elementos pode trazer uma melhora significativa.

Como trabalhos futuros, podemos considerar:

- Experimentos com outros algoritmos de treino para a rede neural.
- Uso da classificação automática para melhorar o sistema de gerenciamento de tarefas de um *grid*.

Referências

[1] B. Behsaz, P. Jaferian, and M.R. Meybodi. Comparison of global computing with grid computing. In *Parallel and Distributed Computing, Applications and Technologies, 2006. PDCAT '06. Seventh International Conference on*, pages 531–534, 2006.

[2] F. Azzedin and M. Maheswaran. Evolving and managing trust in grid computing systems. In *Electrical and Computer Engineering, 2002. IEEE CCECE 2002. Canadian Conference on*, volume 3, pages 1424–1429 vol.3, 2002.

[3] I. Chopra and M. Singh. Analysing the need for autonomic behaviour in grid computing. In *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*, volume 1, pages 535–539, 2010.

[4] T. Jan, M. Piccardi, and T. Hintz. Neural network classifiers for automated video surveillance. In *Neural Networks for Signal Processing, 2003. NNSP'03. 2003 IEEE 13th Workshop on*, pages 729–738, 2003.

[5] Bo Yang and Yuanzhang Sun. An improved neural network prediction model for load demand in day-ahead electricity market. In *Intelligent Control and Automation, 2008. WCICA 2008. 7th World Congress on*, pages 4425–4429, 2008.

[6] K. Fukushima and N. Wake. Handwritten alphanumeric character recognition by the neocognitron. *Neural Networks, IEEE Transactions on*, 2(3):355–365, 1991.

[7] Jinwook Go, Gunhee Han, Hagbae Kim, and Chulhee Lee. Multigradient: a new neural network learning algorithm for pattern classification. *Geoscience and Remote Sensing, IEEE Transactions on*, 39(5):986–993, 2001.

[8] Ahmad Hammad, T. Harenberg, D. Igdalov, P. Mattig, D. Meder-Marouelli, and P. Ueberholz. A job monitoring system for the lcg computing grid. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, page 5 pp., april 2006.

[9] K. Yang, A. Galis, and C. Todd. Policy-based active grid management architecture. In *Networks, 2002. ICON 2002. 10th IEEE International Conference on*, pages 243 – 248, 2002.

[10] Dorin Carstoiu and Alexandra Cernian. A framework for volunteer grid computing management. In *Roedunet International Conference (RoEduNet), 2010 9th*, pages 358 –363, june 2010.

[11] Hao Tian. A new resource management and scheduling model in grid computing based on a hybrid genetic algorithm. In *Computing, Communication, Control, and Management, 2008. CCCM '08. ISECS International Colloquium on*, volume 3, pages 113–117, 2008.

[12] Jeff Heaton. *Introduction to Neural Networks with Java*. Heaton Research, second edition edition, 2008.

[13] Shuai Zhang, Xuebin Chen, Shufen Zhang, and Xiuzhen Huo. The comparison between cloud computing and grid computing. In *Computer Application and System Modeling (ICCASM), 2010 International Conference on*, volume 11, pages V11–72–V11–75, 2010.

[14] K. Qadir, H.F. Ahmad, R. Ur Rasool, S. Mahmood, and H. Suguri. System architecture for efficient grid resource management. In *Intelligent Systems, 2008. IS '08. 4th International IEEE Conference*, volume 3, pages 22–20–22–25, 2008.