XVII Encontro de Jovens Pesquisadores da UCS

Tipo de Bolsa: Voluntário

Utilização de Grids Computacionais para a paralelização de um algoritmo de aprendizado de máquina

Bolsista: Mauricio Adami Mariani - mamarian@ucs.br
Orientador: Sérgio Echeverrigaray
Colaboradores: Günther J.L. Gerhardt, Scheila de Avila e Silva, André Martinotto
Sigla do projeto: Promotores

Introdução e Justificativa

Para que um determinado gene tenha sua expressão realizada de maneira correta, uma sequência anterior a ele deve ser reconhecida. Esta sequência é denominada de promotor. Muitas técnicas de aprendizado de máquina (AM) são empregadas para reconhecer os promotores, dentre elas, as redes neurais artificiais. As redes neurais são uma importante ferramenta, mas em determinados experimentos podem levar muito tempo para processar um resultado, já que necessitam de uma grande quantidade de simulações para produzirem resultados válidos estatisticamente. Uma alternativa para a redução do tempo de processamento é a utilização de *grids* computacionais. *Grids* são um conjunto de computadores processando uma tarefa em comum dividida entre estes, porém não são dedicados, ou seja, esses baseiam-se na utilização do tempo ocioso desses computadores. A ferramenta utilizada será o *Ourgrid*. Entre os principais objetivos está o processamento de tarefas *Bag-of-Tasks*, isto é, as tarefas que são independentes não necessitando de comunicação entre elas.

Objetivo

O objetivo do trabalho é a paralelização do algoritmo de reconhecimento de sequências promotoras através da utilização da ferramenta de *grid Ourgrid*.

Metodologia

Para realizar o presente trabalho, as sequências promotoras foram retiradas do banco de dados RegulonDB, de domínio público, em sua versão de março de 2009, disponível no site http://regulondb.ccg.unam.mx/index.html. As simulações foram realizadas no ambiente R. Este software permite a realização de análise estatística, treinamento de RNs, extração de regras, entre outras funções. Seu uso é intermediado por um algoritmo de manipulação de dados. Os blocos de programação desse *script* foram analisados separadamente.

Inicialmente analisou-se o tempo de cada laço *for (laço de repetição)* do algoritmo:

- O tempo do laço *for* mais externo referente ao número de camadas ocultas da rede neural era de 104.400 segundos, aproximadamente.
- O tempo do subsequente for referente ao número da rede era de : 13050 segundos, aproximadamente.
- O tempo do terceiro for referente ao numero de iterações era de: 1305 segundos, aproximadamente.
- O tempo do último for era de: 130 segundos, aproximadamente.
- E o tempo do bloco de comandos responsáveis pela execução da rede neural era de: 13 segundos, aproximadamente.

Como os laços for não eram dependentes entre si, cada ciclo pode ser executado separadamente sem a espera de seu antecessor. Por esse motivo, foram paralelizados os dois primeiros laços de repetição. O novo algoritmo responsável pela geração do script a ser executado no grid é o seguinte, conforme vê-se na Figura 1:

```
#!/usr/bin/env python

script_grid=open('script_grid.jdf','w+')

script_grid.write('job:\n')
script_grid.write('\t'+'label: Mauricio\n')

for oculto in range(1,9):    #for Noculto in range(2,10)
   for i in range(1,11):
        script_grid.write('task:\n')
        script_grid.write('\t\t'+'init:\t'+'put TreinoRNProm_Sigma24AleatAleat1 TreinoRNProm_Sigma24AleatAleat1\n')
        script_grid.write('\t\t'+'put TreinoRNProm_Sigma24AleatAleat2 TreinoRNProm_Sigma24AleatAleat2\n')
        script_grid.write('\t\t'+'put TreinoRNProm_Sigma24AleatAleat3 TreinoRNProm_Sigma24AleatAleat3\n')
        script_grid.write('\t\t'+'put TreinoRNProm_Sigma24AleatAleat4 TreinoRNProm_Sigma24AleatAleat4\n')
        script_grid.write('\t\t'+'put TreinoRNProm_Sigma24AleatAleat5 TreinoRNProm_Sigma24AleatAleat5\n')
        script_grid.write('\t\t'+'put TreinoRNProm_Sigma24AleatAleat6 TreinoRNProm_Sigma24AleatAleat6\n')
        script_grid.write('\t\t'+'put TreinoRNProm_Sigma24AleatAleat7 TreinoRNProm_Sigma24AleatAleat7\n')
        script_grid.write('\t\t'+'put TreinoRNProm_Sigma24AleatAleat8 TreinoRNProm_Sigma24AleatAleat8\n')
        script_grid.write('\t\t'+'put TreinoRNProm_Sigma24AleatAleat9 TreinoRNProm_Sigma24AleatAleat9\n')
        script_grid.write('\t\t'+'put TreinoRNProm_Sigma24AleatAleat10 TreinoRNProm_Sigma24AleatAleat10\n')
        script_grid.write('\t\t'+'put TesteRNProm_Sigma24Aleat1 TesteRNProm_Sigma24Aleat1\n')
        script_grid.write('\t\t'+'put TesteRNProm_Sigma24Aleat2 TesteRNProm_Sigma24Aleat2\n')
        script_grid.write('\t\t'+'put TesteRNProm_Sigma24Aleat3 TesteRNProm_Sigma24Aleat3\n')
        script_grid.write('\t\t'+'put TesteRNProm_Sigma24Aleat4 TesteRNProm_Sigma24Aleat4\n')
        script_grid.write('\t\t'+'put TesteRNProm_Sigma24Aleat5 TesteRNProm_Sigma24Aleat5\n')
        script_grid.write('\t\t'+'put TesteRNProm_Sigma24Aleat6 TesteRNProm_Sigma24Aleat6\n')
        script_grid.write('\t\t'+'put TesteRNProm_Sigma24Aleat7 TesteRNProm_Sigma24Aleat7\n')
        script_grid.write('\t\t'+'put TesteRNProm_Sigma24Aleat8 TesteRNProm_Sigma24Aleat8\n')
        script_grid.write('\t\t'+'put TesteRNProm_Sigma24Aleat9 TesteRNProm_Sigma24Aleat9\n')
        script_grid.write('\t\t'+'put TesteRNProm_Sigma24Aleat10 TesteRNProm_Sigma24Aleat10\n')
        script_grid.write('\t\t'+'put script_grid.sh script_grid.sh\n')
        script_grid.write('\t\t'+'put Script.R Script.R\n')
        script_grid.write('\t'+'remote: ./script_grid.sh '+ str(oculto) + ' ' + str(i) +'\n')
        script_grid.write('\t'+'final: get arquivo_'+ str(oculto) +'_'+ str(i) +'.tar.gz' + ' arquivo_'+ str(oculto) +'_'+ str(i) +'.tar.gz\n')
script_grid.close()
```

Figura 1. Algoritmo responsável pela paralelização das simulações no *grid* computacional.

Resultados e Conclusões

As simulações realizadas sem a paralelilzação possuiam um tempo total de execução do algoritmo (mesmo tempo do ciclo *for* mais externo) de 29 horas, aproximadamente. Com a utilização da tecnologia de *grids* computacionais o tempo reduziu-se drasticamente para 39 minutos. Assim, permite-se maior agilidade na obtenção e análise de resultados.

A grande vantagem dos *grids* computacionais está na utilização de várias máquinas processando simultaneamente e pela ocupação de máquinas em seu tempo ocioso.

Agradecimentos

O autor gostaria de agradecer ao orientador e co-orientador pelo apoio, à Universidade de Caxias do Sul pela concessão do uso do *grid* computacional da instituição e ao colaboradores pelos esclarecimentos necessários.

Referências

SILVA, S. de A.;Redes Neurais Artificiais aplicadas na caracterização e predição de regiões promotoras;2006;144 p.; Dissertação (Trabalho de Conclusão de Mestrado em Computação Aplicada) – Universidade do Vale dos Sinos (UNISINOS), São Leopoldo.

SANTOS, A. A. da L. dos. Reconhecimento de Padrões em DNA Utilizando Computação Publica. . 80p. Monografia (Trabalho de Conclusão em Ciências da Computação) – Universidade de Caxias do Sul, Caxias do Sul.

ANDRADE, N.;COSTA, L .;GERMÓGLIO, G.;CIRNE, W.; Peer-to-peer grid computing with the OurGrid Community. Universidade Federal de Campina Grande, Campina Grande, 2005.

ANDRADE, N.;CIRNE W.;BRASILEIRO, F.; OurGrid: An approach to easily assemble grids with equitable resource sharing. Universidade Federal de Campina Grande, Campina Grande, 2003.