Otimização do Balanceamento de Carga usando OAR

Eduardo Raug Pinheiro Machado[1,2], Rodrigo Cargnelutti[1], Adriano Petry[1]

[1]Laboratório de Computação para Clima Espacial – Centro Regional Sul de Pesquisas Espaciais – Instituto Nacional de Pesquisas Espaciais – LCCE/CRS/INPE/MCTI
Caixa Postal 5021 – 97.105-970 – Santa Maria – RS – Brasil

[2]Curso de Ciência da Computação – Centro de Tecnologia – Universidade Federal de Santa Maria – CT/UFSM
Av. Roraima nº 1000 – Cidade Universitária – Santa Maria – RS – Brasil

{eduardoraug,cargnelutti,dr.adriano.petry}@gmail.com

1. Introdução

Este trabalho insere-se nas atividades desenvolvidas pelo Laboratório de Computação para Clima Espacial (LCCE) do Centro Regional Sul de Pesquisas Espaciais (CRS/INPE). Basicamente objetiva-se aperfeiçoar um sistema em operação que realiza diariamente a previsão da dinâmica da ionosfera terrestre, baseado em um modelo físico que simula o comportamento e interações físico-químicas que acontecem nessa camada da atmosfera, entre altitudes de 90Km a 1000Km [Pereira et al. 2010].

O balanceamento de carga [Williams 1991] é utilizado para maximizar o desempenho de sistemas paralelizáveis, realizando de forma equilibrada a distribuição da carga de trabalho de acordo com a disponibilidade de recursos. Para se aperfeiçoar o sistema de previsão, foi feita uma otimização do balanceamento de carga via OAR no script que roda diariamente o sistema de previsão. Além disso, foi implementado um novo modo de submissão de processos no OAR utilizando a memória como referência. Essas duas metodologias para otimização de desempenho são chamadas neste artigo de Balanceamento via Script e Balanceamento via Banco de Dados do OAR.

2. O Problema

Para realizar a previsão diária da dinâmica ionosférica, foi utilizado um cluster de computadores em operação no CRS/INPE. No entanto, os nós de processamento disponíveis apresentam configurações de hardware diferentes, podendo ser divididos em 2 grupos distintos. No primeiro grupo existem 2 nós de processamento, cada um contando com 2 processadores Intel Xeon 2.93 GHz, 24 núcleos de processamento e 32 GB de memória RAM. O segundo grupo abrange outros 5 nós de processamento, que apresentam cada um 2 processadores Intel Xeon 2.53 GHz, 16 núcleos de processamento e 16 GB de memória RAM. Outros 2 nós, não utilizados para processamento, disponibilizam 3.6 TB para armazenamento de dados.

As principais atividades a serem executadas pelo sistema, divididas em diversos processos executados paralelamente, requerem a utilização diferenciada de recursos de hardware. Uma dessas atividades, chamada de Assimilação de Dados, exige a utilização maciça de memória, uma vez que requer o carregamento de grandes quantidades de informações, para acesso e processamento. Outra das atividades, a Simulação,

encarrega-se de processar as reações intermoleculares para fornecer a dinâmica temporal em pontos discretos. Essa atividade não utiliza muita memória, mas executa de forma intensiva o processamento de dados. Assim, o problema avaliado neste trabalho refere-se basicamente a otimização da distribuição de recursos de hardware heterogêneos entre atividades com necessidades diferentes em termos de memória e processamento, utilizando um gerenciador de recursos de hardware.

Neste trabalho foi utilizado um gerenciador de recursos livre denominado OAR [Nicolas e Joseph 2011], capaz de gerenciar recursos distribuídos usando banco de dados relacional MySQL ou PostgreSQL. Ele apresentou bons resultados em uma análise comparativa realizada por [Aliaga 2010].

3. Balanceamento via Script

Levando-se em conta os recursos heterogêneos disponíveis e a demanda de processamento e memória diferente em cada uma de suas atividades, uma alternativa para implementar o balanceamento de carga é a submissão dos vários processos de cada atividade de forma diferenciada. Para isso, o script de execução do sistema foi alterado de modo a alocar os recursos também de forma heterogênea, fazendo com que o uso do cluster fosse maximizado, diminuindo assim o tempo necessário para a atividade ser finalizada, e, também, respeitando as limitações de hardware.

O script foi alterado de modo a rearranjar a utilização dos recursos. Primeiramente foram criadas variáveis contendo o limite máximo de carga em função da configuração de hardware. Para cada um dos dois grupos de nós existentes (detalhados no item 2), o valor dessas variáveis foi obtido empiricamente, através de sucessivos testes. A cada submissão é testado se este limite máximo de carga foi alcançado, e a submissão é forçada para que ocorra em um ou outro grupo de nós. A desvantagem dessa abordagem está na necessidade de alteração do algoritmo de submissão conforme a atividade executada e o hardware disponível. O pseudocódigo a seguir ilustra o funcionamento do algoritmo utilizado:

```
1 funcao submissoes (numTotalDeProcessosAseremSubmetidos, X, Y) {
2         numProcessosSubmetidos,processosRodandoGrupo1,processosRodandoGrupo2 <- 0;
3         limiteProcessamentoGrupo1 <- X; limiteProcessamentoGrupo2 <- Y;
4
5         while(numProcessosSubmetidos < numTotalDeProcessosAseremSubmetidos) {
6                 processosRodandoGrupo1 <- verificaNumeroDeProcessosRodando(grupo1);
7                 processosRodandoGrupo2 <- verificaNumeroDeProcessosRodando(grupo2);
8
9                 if (processosRodandoGrupo1 < limiteProcessamentoGrupo1) {
10                        oarSubmete(grupo1);
11                        numProcessosSumetidos <- numProcessosSubmetidos + 1;
12                }
13                else if (processosRodandoGrupo2 < limiteProcessamentoGrupo2) {
14                        oarSubmete(grupo2);
15                        numProcessosSumetidos <- numProcessosSubmetidos + 1;
16                }
17        }
18 }
```

4. Balanceamento via Banco de Dados do OAR

Normalmente, durante o processo de cadastramento de recursos no banco de dados do OAR, é criado um registro para cada núcleo de cada processador disponível. Para isso,

são criados os campos “cpu” e “core” na tabela “resources” do banco, ilustrada na figura 1. Ao executar, cada processo utiliza uma quantidade de recursos pré-estabelecida. No entanto, essa abordagem não é capaz de balancear os processos em função da memória disponível em cada nó. Para viabilizar esse tipo de balanceamento, foi adicionado outro campo, chamado “mem”, na tabela de recursos. Contudo, para o hardware disponível, o primeiro grupo de nós apresenta uma quantidade igual a 32/24 GB de memória por núcleo de processamento, enquanto o segundo tem apenas 1,0 GB por núcleo. Dessa forma, não poderíamos manter o mesmo número de registros na tabela “resources”, pois, ao submeter uma solicitação de memória para um determinado processo, não saberíamos se a quantidade alocada seria múltipla de 1,333 GB ou 1,0 GB.

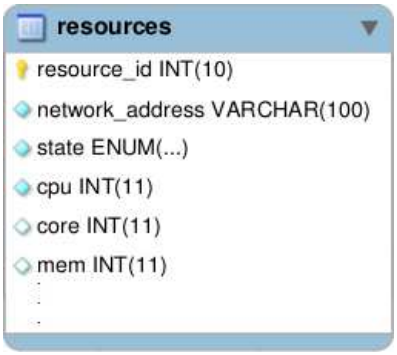

Figura 1. Simplificação da estrutura da tabela “resources”.

Para solucionar de forma genérica a questão da submissão de processos em função da memória disponível, foi estabelecida uma unidade de memória comum entre os dois grupos de nós, capaz de associar a cada núcleo de processamento o valor correto de memória correspondente. Assim, definindo-se uma unidade de memória igual a 1/24 GB, cada núcleo de processamento do primeiro grupo de nós conterá 32 registros na tabela “resources”, associando então 1,333 GB por núcleo. Para o segundo grupo, 24 registros por núcleo associam 1,0 GB. Na submissão de um determinado processo, ao requerer, por exemplo, 3,0 GB para execução, serão solicitadas 72 unidades de memória ao OAR.

5. Resultados

Foram rodadas dez simulações e registrados os tempos de processamento médios para as atividades com alta demanda por memória (Assimilação de Dados) e processamento (Simulação), utilizando as soluções de otimização do balanceamento de carga via script, via banco de dados do OAR e sem otimização de balanceamento (apenas definindo um número de núcleos por processo). Os resultados são mostrados na tabela 1.

Tabela 1. Tempos médios para processamento das Atividades

Solução utilizada	Assimilação de Dados	Simulação
Otimização do balanceamento de carga via script	447 s	658 s
Otimização do balanceamento de carga via banco de dados do OAR	555 s	657 s
Sem otimização do balanceamento de carga	534 s	661 s

6. Conclusão

Para os testes realizados, a otimização do balanceamento de carga via script se mostrou a melhor alternativa para a atividade que requer grande uso de memória (Assimilação de Dados), uma vez que concentrou a execução nos melhores processadores, considerando a memória disponível. A solução via banco de dados limitou o uso dos processadores da mesma forma que a solução via script, porém a distribuição de tarefas foi realizada de forma diferente. Na solução via banco de dados, os melhores processadores (grupo 1) acabaram praticamente não sendo utilizados, pois os demais processadores (grupo 2) foram capazes de concluir os processos recebidos antes que todos os processos tivessem sido distribuídos. Assim, eles acabaram recebendo outros processos para execução, deixando os processadores do grupo 1 ociosos. Isso aconteceu, pois verificou-se que a distribuição de tarefas feita pelo escalonador do OAR procura pelos recursos solicitados considerando a ordem inversa do seu registro no banco de dados. Ou seja, como os processadores do grupo 2 foram inseridos após os processadores do grupo 1, eles foram privilegiados na distribuição de tarefas. Assim, a otimização via banco de dados, apesar de facilitar o processo de submissão, não apresentou um desempenho melhor que a abordagem "sem otimização".

Já para a atividade com alta demanda por processamento (Simulação), não foi verificada uma diferença significativa entre as abordagens em termos de tempo de processamento. Isso ocorreu uma vez que a otimização do balanceamento de carga via banco de dados resultou em uma distribuição dos processos igual a alternativa "sem otimização". A otimização via script, apesar de distribuir os processos de forma a sobrecarregar menos os processadores, não mostrou diferença significativa para as outras abordagens, uma vez que, assim como nas abordagens anteriores, vários núcleos de processamento ficaram ociosos em todos os processadores. É possível que para atividades que requeiram o processamento de uma carga maior as diferenças entre as abordagens sejam mais evidenciadas.

Agradecimentos: Os autores agradecem o auxílio recebido via FAPERGS (bolsa PROBIC processo 11/0434-9) e CNPq (bolsa PCI-DD processo 303460/2011-3).

Referências

Pereira, A. G. ; Petry, A. ; Souza, J. R. ; Charão, A. S. ; Velho, H. F. C. "Aplicação de Técnicas de Paralelismo em Modelo para Previsão da Dinâmica da Ionosfera Terrestre", In: International Symposium on Computer Architecture and High Performance Computing, 2010. p. 13-16.

Williams, R. D. (1991) "Performance of dynamic load balancing algorithms for unstructured mesh calculations", Concurrency: Practice and Experience, 3: 457–481.

Aliaga, A. H. M. (2010) "Estudo comparativo de escalonadores de tarefas para grades computacionais", http://www.ime.usp.br/~alvaroma/mestrado/qualificacao/qualificacao.pdf, Novembro de 2011.

Nicolas, C. and Joseph, E. (2011) "Resource Management System for High Performance Computing", http://oar.imag.fr, Novembro de 2011.