Técnicas de paralelização trivial e alto desempenho em processos MCMC

Jean Paulo Sandri Orengo, Roberto da Silva, Cláudio Geyer

FURG - Fundação Universidade Federal do Rio Grande
Campus Carreiros, (53) 233-6500
ec7jpso@furg.br
UFRGS - Universidade Federal do Rio Grande do Sul
Instituto de Informática,(51) 3316-6165
{rdasilva,geyer}@inf.ufrgs.br

Introdução

Alguns modelos vastamente conhecidos traduzidos por métodos Monte Carlo representam bons testes, devido ao alto custo computacional de processamento, para avaliação do comportamento e do desempenho de plataformas de computação que se utilizem de infra-estrutura de equipamentos de uso compartilhado.

O objetivo deste texto não é propor modelos ou resultados novos para a Física de muitas componentes, o intuito é unica e exclusivamente usar um específico algoritmo método Monte Carlo como uma interessante aplicação para testes de um software de paralelização do tipo *bag of tasks* desenvolvido para ambientes distribuídos e compartilhados no contexto de computação de alto desempenho.

Esta aplicação, um "velho e conhecido modelo" vastamente utilizado pelos físicos com sucesso, e proposto por Ernst Ising em 1902 para descrever o comportamento coletivo de um ferromagneto, ficou conhecido por modelo de Ising, e pressupõe que um conjunto de variáveis aleatórias as quais simulam os spins, evoluem estocasticamente segundo um processo de Markov de acordo com a prescrição do algoritmo de Metrópolis [N.ME 53].

Aplicações que utilizam o método Monte Carlo, muitas vezes, permitem a decomposição do trabalho a ser processado em unidades de trabalho (UTs) independentes. Neste trabalho a aplicação foi modelada na forma de *mestre/trabalhador*, aproveitando parte da estrutura oferecida pelo projeto ISAM[1], em desenvolvimento no Instituto de Informática da UFRGS, mas específicamente o sub-projeto EXEHDA-MW. Esta infra-estrutura vem sendo utilizada para execução de problemas em ambientes de "grid computing".

O EXEHDA-MW

No EXEHDA-MW o mestre é a entidade responsável pelo particionamento do processamento como um todo em UTs, pela distribuição destas UTs, e pela reunião dos resultados parciais obtidos, objetivando a obtenção do resultado final.

[1] http://www.inf.ufrgs.br/~isam

A transferência das UTs ocorre segundo uma política "receiver-initiated", na qual cada trabalhador é responsável por solicitar UTs do mestre. Os trabalhadores operam segundo um algoritmo bastante simples: solicitam trabalho, processam o trabalho gerando resultados parciais, enviam os resultados para o mestre e voltam a solicitar trabalho até que não exista mais nada a processar.

O mestre é uma entidade um pouco mais complexa. Ele é subdividido em alguns módulos de controle: (i) **Splitter** – é o módulo responsável por particionar o trabalho em UTs; ele recebe o tamanho desejado de UT, permitindo que o tamanho da UT seja ajustado individualmente conforme o poder computacional dos trabalhadores, permitindo desta forma que a granulosidade do problema seja ajustada; (ii) **WU Manager** – gerencia as UTs, mantendo informações de quais precisam ainda ser processadas, quais estão sendo processadas e quais já foram processadas. Além disto, ele é responsável pela manutenção dos resultados parciais das UTs; (iii) **Aggregator** – reune os resultados parciais para a geraç ão do resultado final.

Esta arquitetura está preparada para execução em plataformas heterogêneas, todos módulos foram construídos em Java. A arquitetura também contempla duas abordagens com relação ao tamanho das tarefas: (i) **UTs de tamanho fixo** – nesta abordagem, abreviada por UTF, o mesmo tamanho de UT é entregue para cada um dos trabalhadores, sem que seja levado em consideração o poder de processamento disponível em cada trabalhador; (ii) **UTs de tamanho ajustável** – nesta abordagem, abreviada por UTA, o tamanho das UTs é ajustado dinâmicamente pelos trabalhadores [YAM 2003]. Cada trabalhador recebe uma meta de tempo para processamento da UT. Conforme o tempo de execução obtido e a meta de tempo, um novo tamanho para a próxima UT é calculado pelo trabalhador, buscando atingir a meta na próxima execução.

A área de “grid computing” [FOS 99] oferece uma alternativa real de processamento de alto desempenho utilizando recursos distribuídos geograficamente. Esta proposta apresenta uma série de desafios para execução de aplicações paralelas e distribuídas. Nacional e internacionalmente já é possível encontrar trabalhos que tratem de algumas questões relacionadas a esta nova área [YAM 2003, TEA 2001].

Dois aspectos que dificultam a obtenção de bom desempenho em aplicações distribuídas no contexto de *grid computing* são: (i) a elevada heterogeneidade dos recursos envolvidos; e (ii) o regime de uso compartilhado que, em geral é empregado. Estratégias de balancemento de carga podem ser empregadas no sentido de minimizar o impacto negativo que pode ser causado por computadores de baixo poder computacional ou sobrecarregados. Na próxima secção falaremos um pouco sobre o modelo e alguns detalhes sobre quais e como foram as medidas tomadas.

A aplicação e seu contexto na Ciência da Computação

Essencialmente o modelo de Ising é um caso simples de um MCMC (Monte Carlo Markov Chain). Sua implementação procede da seguinte maneira; considera-se um conjunto de N variáveis aleatórias $\sigma = \{\sigma_i\}_{i=1}^{N}$, que podem assumir somente dois possíveis valores $\sigma_i \in \{\pm 1\}$ dispostas em um reticulado $d-$dimensional. Essas variáveis modelariam os "spins"dos N ions magnéticos de um material ferromagnético (para a nossa implementação $d = 2$ e $N = L^2$).

A energia de interação se dá somente entre dois "spins"vizinhos mais próximos $\langle i,j\rangle$, de forma que dois vizinhos contribuem com um valor $-J$ se eles possuem o mesmo sinal e $+J$ se possuem sinas contrários, onde $J > 0$ de forma que a energia total do sistema é expressa por $E = -J\sum_{\langle i,j\rangle}\sigma_i(x_i,y_i)\ \sigma_j(x_j,y_j)$, onde por comodidade e sem perda de generalidade faremos $J = 1$, com $(x_k,y_k) \in \{1,...,L\}^2$.

A transição de uma específica variável local σ_i para um estado σ_j, denotada por $\sigma_i \to \sigma_j$ se dá com probabilidade igual à $\min\{1,\exp(-\beta\Delta E)\}$, onde $T = 1/\beta$ é a temperatura a qual o sistema está submetido, e $\Delta E = -2\sigma_i(x_i,y_i)\cdot S_i$ é a variação de energia total resultante da transição local $\sigma_i \to -\sigma_i$, onde $S_i = \sigma_i(x_i+1,y_i)+\sigma_i(x_i-1,y_i)+\sigma_i(x_i,y_i-1)+\sigma_i(x_i,y_i+1)$ é a soma das 4 variáveis de spin vizinhas mais próximas a $\sigma_i(x_i,y_i)$, os únicos responsáveis pela transição de acordo com o modelo. Quando todos sítios (x_i,y_i), sequencialmente, são percorridos nós dizemos que uma unidade de tempo (um passo de Monte Carlo foi executado).

A proposta fundamental quando se estuda algoritmos deste tipo é que para $T < \infty$, variáveis termodinâmicas são expressas como médias simples sobre passos de Monte Carlo e são tomadas a partir de um tempo t_∞ (conhecido como tempo de equilibração) na qual o sistema atinge uma distribuição invariante como é característico de processos de Markov que satisfazem o balanço detalhado.

O fato é que a partir de t_∞, o sistema é representado por uma matriz de equilíbrio $[\sigma^{eq}(x_i,y_j)]_{x_i,y_j=1,...,L}$, composta por elementos ± 1, e as grandezas termodinâmicas, funções das variáveis de spins (isto é das componentes desta matriz) variam estocásticamente em torno de um valor médio como é o caso da energia, da magnetização e calor específico por exemplo, nesta situação de equilíbrio médias aritméticas sobre os diferentes passos de Monte Carlo, representam as médias sob as inúmeras configuracões do sistema, o que seria impraticável obter uma vez que o número de configurações possíveis em sistemas de tamanho N, é exponencial (2^N), o que mostra a grande vantagem de tal abordagem.

Assim a maior dificuldade computacional de realizarmos simulações Monte Carlo, para grandes entradas em problemas deste tipo, é o fato de t_∞ crescer exponencialmente com o tamanho do problema. Este tempo necessário para que o sistema atinja o equilíbrio, torna proibido a simulação de sistemas com muitas componentes, já que o tempo necessário para começar a se tomar as médias sobre os passos de Monte Carlo começa a se tornar impraticável, pois a complexidade do problema torna-se considerável hajavisto que ela seja $O\left(L^2\left[t_\infty + N_u\right]\right)$, onde $N_u = N_{\max}^{(MC)} - t_\infty$, são os passos de Monte Carlo realmente utilizados para se calcular as grandezas supracitadas.

Resultados e Considerações Finais

Foram executadas simulações Monte Carlo para um conjunto de parâmetros fixos. O número de passos utilizados foram $N_{\max}^{(MC)} = 10000$, dispensando um total de $t_\infty = 3000$ passos, sendo que um valor de semente para cada "trabalhador" fora utilizado e para um sistema de dimensão linear $L = 160$, e uma temperatura de $\beta = 1$. O algoritmo toma medidas de 10 em 10 passos após t_∞, uma vez que é interesante para a aplicação descorrelacionar estatisticamente duas configurações sucessivas.

Inicialmente, parte do processamento, as simulações desde $t = 1$ até $N_{\max}^{(MC)}$ eram feitas de forma redundante por cada um dos nodos envolvidos. Observou-se porém, que

t_∞	nodos	abordagem	tempo
com redundância	4	UTF	2min 48.123s
sem redundância	4	UTF	2min 37.392s
			2min 39.060s
			2min 37.595s
sem redundância	5	UTA	1min 9.586s
sem redundância	6	UTA	1min 10.732s
			1min 04.456s
			1min 04.035s

Tabela 1: Medidas de tempo usando o EXEHDA-MW. O algortimo foi utilizado primeiro executando o tempo de equilibrio em todos os nodos e depois somente no mestre.

este processamento levava a uma mesma situação em todos os nodos e optou-se pelo processamento desta etapa no mestre. Na tabela 1 são apresentados os resultados para as execuções "com redundância" e "sem redundância", relativas a cada uma das situações descritas. Embora pequeno, observa-se que houve um ganho em termos de tempo de processamento.

Para obtenção dos resultados foram utilizadas as duas abordagens fornecidas pelo EXEHDA-MW, conforme apresentado na seção referente ao EXEHDA-MW, para determinação dos tamanhos das tarefas. Os resultados obtidos com a estratégia UTA apresentaram-se melhores do que os resultados utilizando a estratégia UTF. Este fato ocorre devido a diferença de poder computacional entre os equipamentos, a utilização de tamanho de tarefa fixa implica em um custo maior na barreira de sincronização ao final da execução, quando ficam os nodos mais rápidos sem trabalho e os nodos mais lentos processando suas últimas tarefas. Com o uso da estratégia de tamanho de tarefa ajustável este problema é minimizado, pois o tamanho das tarefas fica atrelado ao tempo de resposta de cada nodo, limitando assim o atraso máximo na barreira de sincronização. Nestas simulações foram utilizados os seguintes tipos de microprocessadores: um Pentium II 300 MHz, um Pentium IV 1GHz, demais Pentium III 900 MHz.

Referências

[FOS 99] FOSTER, I.; KESSELMAN, C. (Eds.). **The grid**: blueprint for a new computing infrastructure. San Francisco, CA: Morgan Kaufmann, 1999.

[N.ME 53] N. METROPOLIS A. ROSENBLUTH, M. R. A. T. e. E. T. Equations of state calculations by fast computing machines. **Journal of Chemical Physics**, v.21, p.1087–1092, 1953.

[TEA 2001] TEAM, S. **Search for extraterrestrial intelligence**. http://setiathome.ssl.berkeley.edu/ (acesso em outubro de 2001).

[YAM 2003] YAMIN, A. C. et al. Towards merging context-aware, mobile and grid computing. **The International Journal of High Performance Computing Applications**, v.17, n.2, p.191–203, 2003.