Aplicação de Algoritmos de Otimização para a Simulação Distribuída de Arquiteturas

Joel Giordani Pereira, Rafael Ramos dos Santos, João Carlos Furtado

Departamento de Informática – Universidade de Santa Cruz do Sul (UNISC)
Av. Independência, 2293 – 96.815-900 – Santa Cruz do Sul – RS – Brasil

joelgp@gmail.com, rsantos@unisc.br, jcarlosf@unisc.br

1. Introdução

Para validar o desempenho de uma arquitetura superescalar o uso de simuladores é essencial durante a fase de projeto devido à complexidade da arquitetura. É um processo que envolve simulações e análise dos resultados, que servem como base para novas configurações até que o objetivo seja atingido. O objetivo depende da finalidade do projeto, seja relacionando um custo em área, consumo ou monetário com o desempenho. Este processo envolve conseqüentemente inúmeras simulações através de um processo de refinamento sucessivo. Obter a melhor configuração executando uma simulação para cada configuração de arquitetura possível se torna impraticável pelo tempo envolvido na execução de todas as simulações e pelo volume de recursos computacionais geralmente envolvidos.

Para automatizar este processo e diminuir o tempo de obtenção do resultado, este trabalho propõe a utilização de conceitos de hardware evolutivo, através da implementação de um algoritmo genético para a otimização do processo de busca. O processamento das simulações foi distribuído para maior ganho de desempenho.

2. Otimização da busca de configuração

A simulação de um processador superescalar foi realizada pelo programa sim-outorder, da suite SimpleScalar [BURGER e AUSTIN 1997]. A busca da configuração de uma arquitetura foi automatizada através de hardware evolutivo [THOMPSON 1997], onde sistemas evolucionários são usados projeto de hardware.

Para a validação do algoritmo genético[VIANNA 1998] foram realizadas simulações onde somente os valores de configuração da memória *cache* nível 1 de instruções foram alteradas para gerar 399 configurações diferentes. O objetivo destas simulações era obter o custo/benefício entre a área usada e o desempenho obtido. Para tal, para cada configuração da memória foi relacionada um valor de custo, usado no cálculo do *fitness* do algoritmo genético.

Cada configuração foi simulada usando 5 programas para *benchmarks* diferentes, executando 500 milhões de instruções, ignorando as primeiras 100 milhões. As primeiras 100 milhões de instruções foram ignoradas para que a análise dos resultados do simulador concentrem-se nas instruções responsáveis pela funcionalidade do *benchmark*, sendo que as instruções iniciais geralmente geralmente tratam-se de inicialização de variáveis e leitura de parâmetros. Foram executadas 500 milhões de instruções por representarem a funcionalidade do *benchmark* sem consumir um tempo excessivo de

simulação, pois quanto maior o número instruções executadas pelo simulador, maior o tempo de simulação. No total, foram executadas 1995 simulações, onde foi possível obter o resultado que melhor atende o objetivo das simulações dentro do espaço de busca.

Foi usado uma implementação simples do algoritmo genético usando a linguagem Java, sem otimizações específicas, com a população de oito indivíduos, executando por cinco gerações, percentual de mortalidade 75% e percentual de mutação de 100%. O resultado foi de uma redução no tempo de execução de 90,98% e o melhor valor encontrado pelo algoritmo genético representa 97% do melhor valor encontrado através da execução de todas as possibilidades de configuração.

Por representar maior tempo de execução dentro do processo realizado em cada geração do algoritmo genético, as simulações usadas no cálculo do *fitness* dos indivíduos são executadas paralelamente utilizando-se de distribuição em grid ou Cluster. Cada simulação executada é submetida à fila de processos do Grid ou do Cluster e o resultado retorna para o algoritmo genético usando *sockets* na distribuição em Cluster e usando RMI na distribuição em grid.

Para a realização dos testes, a quantidade de *benchmarks* usados foi reduzida para 2. Com isso, em cada geração do algoritmo genético são executados no máximo 16 simulações simultâneas.

Nos testes usando o cluster Labtec da UFRGS, foram usados 16 nodos e o ganho de desempenho pela distribuição foi de 90,13% em comparação à execução seqüencial do algoritmo genético. Nos testes distribuição em Grid, usando a ferramenta OurGrid [CIRNE 2003] nos computadores do laboratório de Informática da UNISC, também foram usadas 16 estações e o ganho de desempenho pela distribuição foi de 91,30% em comparação à execução seqüencial do algoritmo genético.

3. Conclusão

Considerando os resultados da validação do algoritmo genético, o tempo de execução diminuiu 90,98% e a qualidade da configuração obtida foi muito próxima da ideal, com resultados acima de 97%, usando como parâmetro o melhor resultado.

Em relação à distribuição do processamento do algoritmo genético, em todos os testes e formas de distribuição, o resultado do ganho em desempenho teve resultados muito bons, com valores superiores a 80%.

Referências

Burger, D. C.; Austin, T. M. (1997) The SimpleScalar Tool Set, Version 2.0. Madison: University of Wisconsin, Technical report. (CS-TR-1997-1342)

Cirne, W et. al. (2003) Grid Computing for Bag of Tasks Applications. Proceedings of the Third IFIP Conference on E-Commerce, E-Business and E-Goverment.

Thompson, Adrian. (1997) Evolutionary Robotics: From intelligent robots to artificial life (ER'97) AAI Books. p. 101-125

Vianna, Gerardo Valdisio Rodrigues. (1998) Meta-heurísticas e programação paralela em otimização combinatória. Fortaleza: EUFC.