Trabalho 1 - Resumo do Artigo: Automatic Workload Generation for System-Level Exploration Based on Modified GCC Compiler.

Citação completa: Jari Kreku, Kari Tiensyrja, and Geert Vanmeerbeeck. 2010. Automatic workload generation for system-level exploration based on modified GCC compiler. In *Proceedings of the Conference on Design, Automation and Test in Europe* (DATE '10). European Design and Automation Association, 3001 Leuven, Belgium, Belgium, 369-374.

Objetivo e importância

Este artigo apresenta uma técnica baseada na modificação de um compilador (**GCC**) que permite gerar automaticamente **modelos de "workload"** para a simulação do desempenho de arquiteturas candidatas ao nível do sistema (particularmente em sistemas embebidos). No contexto atual em que há sistemas complexos que executam seus programas tanto sequencialmente quanto concorrentemente, isto é importante porque se precisam de avaliações ou simulações do desempenho rápidas e eficientes que permitam evitar erros críticos do desenho. Porém, neste trabalho a facilidade de geração e "debugging" teve mais prioridade que o tempo de simulação do desempenho, assim melhorar isto é um trabalho futuro para o desenvolvimento de ABSINTH.

Modelos em "System-level exploration", ABSOLUT e ABSINTH

Os "workloads" são abstrações das aplicações ("benchmarks") que se executam num determinado computador ou arquitetura que servem para medir seu desempenho. Esta abstração é a diferença do enfoque deste trabalho com outros, porque aqui o modelo de "workload" imita realmente as estruturas de controle das aplicações, mas os dados de nível mais baixo são presentados como "traces", e também a plataforma de execução é modelada no nível de transação e não de instruções. É importante ressaltar que o artigo se descreve no contexto de outros modelos e enfoques desenvolvidos previamente (ABSOLUT).

O enfoque **ABSOLUT** abstrai os modelos de aplicações como modelos de "workload", estes são assignados aos modelos de execução na plataforma (que são abstraídos como modelos de capacidade) e simulados num nível de transação para obter resultados do desempenho. Os **modelos de "workload"** tem uma estrutura hierárquica, dividindo-se em aplicações e controle comum de concorrência (implementado em C++ e baseado em SystemC), e subsequentemente em: processos, funções, bloques básicos e "branches", e operações primitivas (as operações primitivas podem ser "read", "write" ou "execute").

O **modelo da plataforma** compreende a descrição de HW e SW para a simulação. Este tem três camadas: de componente (inclui processamento, armazenamento e comunicação), de subsistema (instancia componentes e seus conexões) e de arquitetura da plataforma (conecta os modelos de "workload" com as plataformas mediante interfaces). Na fase de alocação se assigna a cada entidade do modelo de "workload" um recurso da plataforma (que poder ser um processador). As saídas da simulação são o tempo de execução e outras medidas do desempenho (como utilização de componentes, operações de I/O, etc.) que permitem decidir se a capacidade de execução da plataforma deve ser melhorada ou a aplicação deve utiliza-la mais eficientemente.

ABSINTH é a ferramenta baseada no enfoque ABSOLUT que permite gerar modelos "workload" automaticamente tendo como entrada um código fonte. ABSINTH foi implementado como extensão de GCC e tem dois passos: o primeiro é **pass_absinth_cf**, o qual constrói a camada de funções do modelo de "workload" (fluxo de controle entre blocos básicos); e o segundo passo é **pass_absinth_bbs**, o qual atravessa o RTL (código intermédio de GCC) para extrair as operações primitivas ("read", "write" e "execute").

Para gerar o modelo de "workload" precisa-se de três fases: primeiramente, o código fonte deve ser compilado com "profiling" (isto ajuda a conhecer as probabilidades de "branches"); depois, o código binário gerado deve ser executado com um conjunto de dados adequado; e finalmente, o código fonte deve ser compilado novamente, mas além de "profiling" com ABSINTH habilitado. Um modelo de "workload" é gerado por cada função do código fonte, assim os modelos podem conter chamados a outras funções de "workload", então é preciso de um **ABSINTH manager** o qual detecta dependências e modifica os arquivos de tal maneira que sejam compilados corretamente para a simulação.

Casos de exemplo

O ABSINTH foi usado para criar modelos de "workload" do codificador de vídeo x264, e duas versões (paralela e sequencial) de um codificador JPEG. O modelo de plataforma consistiu em quatro nodos ARM conectados por roteadores, em que cada um tinha um ARM9 CPU, memoria SRAM, um barramento compartilhado e uma interface a os outros nodos.

Seguiram-se os três passos descritos anteriormente para gerar os modelos de "workload" de x264, obtendo-se um normal e outro com blocos básicos simples. Cinco simulações foram executadas com ambos os conjuntos, as velocidades das simulações foram 1 /70 e 1 /17 do tempo real respectivamente, com o segundo conjunto o tempo foi cerca de 10 segundos menos devido à menor utilização de SRAM. O segundo caso de exemplo foi gerar modelos de "workload" para JPEG seguindo também os três passos; simularam-se quatro conjuntos de modelos de "workload", o primeiro foi sequencial e os outros três foram versões paralelas que foram criadas usando MPA (uma ferramenta para mapear eficientemente aplicações em plataformas "multicore"). O tempo de execução do sequencial foi 70 ms e dos paralelos 55, 53 e 38 ms.