Paralelização Automática de Código em Plataformas Heterogêneas [1]

Título:

Paralelização Automática de Código em Plataformas Heterogêneas [1]

Objetivo:
Prover mecanismos para o uso de paralelização automática em sistemas heterogêneos: Uma abordagem de detecção e extração de paralelismo automaticamente em código legado torna-se interessante uma vez que grande parte das abordagens existentes exigem a intervenção do usuário reescrevendo versões do código para as novas plataformas ou anotando o código para guiar o compilador na tradução das regiões consideradas paralelizáveis. Além disso, fazer uso dos recursos disponíveis por meio de cooperação de elementos heterogêneos e não usar esses recursos de maneira mutuamente exclusiva. Utilização efetiva de recursos e de todos os elementos de processamento do sistema heterogêneo: As plataformas modernas tornam-se cada vez heterogêneas, sendo compostas pelos mais variados elementos de processamento, assim a exploração e uso efetivo de recursos disponíveis em computações torna-se o caminho natural. Gerar código para um ambiente de execução multicore e multiGPU: A parte do trabalho que diz respeito à ferramenta de compilação tem muitas sobreposições com ferramentas como o KernelGen, porém pretendemos avançar na geração de código para plataformas heterogêneas de maneira que combinem esses elementos em um modelo de execução \textit{multicore e multiGPU, sustentando a ideia de particionamento e escalonamento de tarefas para executar aplicações neste ambiente considerando o uso eficiente de recursos disponíveis. As principais contribuições esperadas para o trabalho são: Proposição de um método e de mecanismos para o uso de paralelização automática em sistemas heterogêneos: No contexto desta proposta existem diversos trabalhos e ferramentas em todos os pontos da cadeia indo de geração de código ao escalonamento de tarefas. Nossa proposta busca, por meio da integração de projetos existentes que resolvem subproblemas, montar um ambiente de execução que tenha como entrada aplicações com regiões paralelizáveis em código intermediário que possam ser compiladas para serem executadas em múltiplos dispositivos. Consideramos que são contribuições válidas a criação de novas soluções utilizando como componentes outras soluções existentes que resolvem uma parte do problema. A maioria das abordagens trabalham de forma isolada, não considerando a continuidade ou a combinação de soluções existentes como componentes para criar outras soluções válidas. Esse comportamento acaba diluindo o esforço que poderia trazer avanços para a área em questão, e mantendo as ferramentas em níveis de resultados parecidos. Desenvolvimento de um ambiente de execução capaz de escalonar tarefas simultâneas em um ambiente misto multicore e multiGPU: Buscamos contribuições voltadas aos mecanismos de escalonamento de tarefas para contextos mistos que utilizem de maneira eficiente os recursos disponibilizados por essas plataformas modernas. É sabido que devido à especificidade de muitas aplicações, não há um grau de paralelismo alto que possa ser

explorado. Desta forma, nossa proposta traz a ideia de carregar múltiplas aplicações para que seja possível o escalonamento de tarefas simultâneas, na intenção de se obter o uso efetivo de recursos e explorando o paralelismo natural de aplicações distintas. Explorar recursos e características como memória compartilhada para cooperação entre tarefas: Algumas das ferramentas existentes geram código para GPU, porém seus modelos de computação não utilizam novas possibilidades e recursos que tem sido disponibilizados no decorrer da evolução das arquiteturas das GPUs. Características como Paralelismo Dinâmico e possíveis compartilhamentos de memória, bem como acesso pareado entre GPUs não tem sido utilizadas, e pretendemos explorar essas características. Contribuições com a implementação de novas transformações de laços: Iremos usar as transformações de laços que estiverem disponíveis no PoLLy, mas se for necessário criar novas como a kernelize que seria transformar o código de funções em linguagem intermediária pertencentes a um módulo com os identificadores necessários para a execução da função como kernel nos dispositivos aceleradores.

Descrição:

O projeto propõe o estudo e a aplicação de técnicas e ferramentas de paralelização automática no desenvolvimento de uma ferramenta de compilação e de um ambiente de execução. A ferramente de compilação irá detectar regiões paralelizáveis que possam ter sua execução acelerada e o ambiente de execução irá tratar do escalonamento de tarefas de múltiplas aplicações, explorando a integração entre multicore e multiGPU em um modelo de paralelização que trabalhe tanto com paralelismo de distribuição de computação, quanto cooperação entre os elementos de processamento na resolução de problemas. No cenário atual é evidente a grande distância entre o potencial de desempenho disponibilizado pelas plataformas modernas e as aplicações que precisam fazer uso desse potencial. De um lado temos plataformas multicore que integram arranjos de coprocessadores e GPUs com grande poder de processamento, e do outro aplicações científicas com código legado que continuam sendo utilizadas e servindo aos propósitos para os quais foram criadas. Um dos principais desafios para essas plataformas modernas é a coexistência com outras plataformas, com outras tecnologias já consolidadas e com suas respectivas linguagens de desenvolvimento amplamente utilizadas.

Programar novas aplicações ou traduzir aplicações de código legado para estas novas plataformas não é uma tarefa trivial. Os kits de desenvolvimento, ainda não estão maduros o suficiente para serem acessíveis de maneira amigável a todas as classes de usuários. Os kits fornecem aos desenvolvedores bibliotecas do ambiente de execução (runtime) e dos drivers que abstraem o acesso aos recursos do hardware até certo nível. Quanto a forma de geração de código em grande parte dos trabalhos relacionados, as ferramentas consideram código escrito na linguagem C ou Fortran, e a partir dele geram a versão do código paralela com o apoio dos compiladores disponibilizados pelos fabricantes ou com outras ferramentas com backends que geram o código final para GPU. No caso da abordagem de diretivas de compilação a ideia seguida é a de anotar o código com diretivas (\#pragma) que na fase de preprocessamento são tratadas e é feita a inserção de definições e macros no código que no processo de ligação com o ambiente de execução são substituídas pelo código que fará o chaveamento entre dispositivos.

Na abordagem de paralelização automática o que é diferente é que o código não precisa ser anotado, as regiões candidatas à paralelização são detectadas automaticamente e da mesma forma código para a ligação com o ambiente de execução é inserido no código final. Estudos preliminares mostram que as ferramentas existentes e abordagens conhecidas até o

momento, não exploram efetivamente todos os recursos disponíveis nessas plataformas. Não fazem uso eficiente dos recursos ou geram código sem a preocupação com essa questão.

Consideram na geração de código apenas uma das classes de dispositivos, gerando código para uma única CPU ou GPU. Se a opção for gerar código para GPU utilizam a que for especificada pelo usuário ou por padrão escolhem a que tem o maior número de núcleos. De forma análoga, quando o alvo são CPUs ignoram a existência de multicore, gerando código para ser executado em uma única thread, não aproveitando os múltiplos núcleos. Em ambos os casos, não consideram a existência de múltiplos elementos de processamento e nem a combinação deles em uma solução híbrida. Esse comportamento evidencia o uso não eficiente de recursos, que são subutilizados ou desprezados na geração de código e em sua execução, e reforça a ideia de que existe a necessidade de um ambiente de execução que explore cenários heterogêneos, com o uso misto de elementos de processamento multicore e multiGPU, o que abre espaço para o desenvolvimento desta proposta.

Característica:
Ferramenta de Compilação e Ambiente de Execução; Linux.
Início da pesquisa:
terça-feira, 16 Agosto, 2011
Equipe:
Rogério Aparecido Gonçalves, UTFPR e IME-USP; Alfredo Goldman, IME-USP;
Apoio:
Fundação Araucária, Secretaria de Estado da Ciência, Tecnologia e Ensino Superior (SETI-PR) do Governo do Estado do Paraná, DCC do IME-USP, Programa de Pós-Graduação em Ciência da Computação (PPGCC) - CAPES-001-22/2011/CAA III/CGAA/DAV de 30/03/2011.
Licença:
University of Illinois/NCSA Open Source License
Página do Projeto:
http://napsol.icmc.usp.br [2]
Localização:
IME-USP
Categoria(s):
Software
Estado:
Ativo

Feeds