Balanceamento de Carga em Sistemas Paralelos Heterogêneos Baseados em GPU

Anderson Uilian Kauer, Mozart Lemos de Siqueira

UNILASALLE – Centro Universitário La Salle
Av. Victor Barreto, 2288, Centro – 92010-000 – Canoas – RS – Brasil

andersonkauer@gmail.com, mozarts@unilasalle.edu.br

1. Introdução

A crescente demanda por processamento fez com que novas técnicas de aproveitamento computacional fossem desenvolvidas. Atualmente a maneira mais eficaz para se obter aumento na capacidade processamento é a utilização de mais processadores em uma mesma unidade computacional, no entanto, o consumo energético é diretamente proporcional à quantidade de unidades processamento envolvidas [Wang and Ren 2010].

Com a popularização das placas gráficas (ou GPUs), novas estratégias de aproveitamento computacional tornam-se alvo de muitas pesquisas principalmente no que se refere a abordagens sobre utilização eficiente e balanceamento de carga. A computação heterogênea baseada no uso de GPUs tornou-se foco de muitas dessas abordagens e surge como alternativa viável para computação de alto desempenho. GPUs apresentam consumo de energia inferior comparados com CPUs tradicionais.

GPGPU ou GPUs para propósito geral é um tema bastante amplo e tem sido impulsionado principalmente pelas disponibilidades de bibliotecas específicas desenvolvidas pelos fabricantes, entretanto, a comunicação entre CPUs e GPUs ainda é limitada pelo gargalo do barramento PCIe. Tornando-se necessária a utilização de técnicas específicas de balanceamento de carga para tornar o processamento eficientemente distribuído em ambientes de computação heterogênea baseados em GPUs.

A principal contribuição deste projeto está associada à investigação sobre o balanceamento dinâmico de carga em ambientes de computação heterogênea integrados de CPU e GPU, o que existe atualmente e o que ainda precisa ser melhorado.

2. HCE - *Heterogeneous Computing Environments*

Atualmente, a capacidade de processamento das GPUs é significativamente superior a CPUs convencionais com múltiplos núcleos [Daga; Aji and Feng, 2011]. As GPUs são especializadas em computação intensiva altamente paralela, pois têm mais núcleos de processamento dedicados a dados ao invés de cache e controle de fluxo [Wang *et al.* 2008].

O surgimento de GPUs como unidades de processamento auxiliar tem aumentado o nível de heterogeneidade nas plataformas atuais tanto na arquitetura quanto nos modelos de programação. No entanto, a programação *multi*-GPU herdou alguns dos problemas conhecidos na programação paralela convencional, como por exemplo, o

problema de balanceamento de carga entre as GPUs na resolução de problemas [Acosta *et al.* 2010].

Outro fator importante que deve ser considerado é o gargalo de comunicação em HCE para transferência de dados entre a GPU e CPU através de barramentos PCIe (*Peripheral Component Interconnect express*) o que levou ao projeto de Unidades de Processamento Acelerado (APU - *Accelerated Processing Units*) da AMD, que combinam CPUs e GPUs em um único chip isso implica em algumas vantagens: computadores mais baratos, melhor desempenho e baixo consumo de energia [Doerksen *et al.* 2011]. Nestas arquiteturas, os núcleos CPU e os núcleos programáveis GPU compartilham um caminho comum à memória do sistema. Assim, transferências de dados não atingem um barramento externo do sistema, reduzindo o impacto da lentidão PCIe [Daga; Aji and Feng, 2011].

Muitas pesquisas indicam que a GPUs podem resolver problemas de computação intensiva com grande vantagem comparados com a CPUs *multi-core*, no entanto, devido ao uso especial da GPU, é impossível completar todas as tarefas de computação apenas na GPU. Operações de controle, por exemplo, precisam ser realizados na CPU [Wang *et al.* 2008]. No entanto, o desempenho deste tipo de sistema é muito limitado devido à dependência que existe entre o código e a arquitetura paralela, prejudicando a eficiência do processo de atribuição de tarefas se os programas não forem adaptados para o ambiente heterogêneo [Acosta *et al.* 2010].

3. Balanceamento de Carga em HCE

A forma mais simples de se obter balanceamento de carga, em um cenário onde as tarefas que exigem tempos diferentes para a conclusão devem ser distribuídas igualmente entre os processadores é um problema NP-completo. No contexto da computação heterogênea, os componentes distribuídos do algoritmo de balanceamento de carga devem cooperar na tomada de decisões [Maheshwar 1996].

Três dos cinco maiores supercomputadores mais rápidos do mundo empregam GPUs [Top500 list 2011]. É importante observar que os supercomputadores equipados com GPUs atingem apenas 50% do seu desempenho de pico teórico, enquanto os supercomputadores sem GPUs atingem 78% do pico teórico. Isto demonstra que há certos aspectos da GPUs que limitam o desempenho para o *Linpack*, *benchmark* utilizado para classificar os supercomputadores na Top500.

Em sistemas acelerados por GPUs, as GPU não só proporcionam um alto desempenho de pico, mas também reduzem o consumo de energia consideravelmente. Entretanto, devido à sua limitação arquitetônica, o desempenho esta relacionado com a intensidade de aplicação do cálculo, paralelismo e modelo de acesso à memória, etc. Portanto, a distribuição de trabalho razoável entre CPUs e GPUs é uma forma eficaz para explorar melhor os sistemas heterogêneos CPU-GPU [Wang and Ren 2010].

4. Trabalhos Relacionados e Resultados Esperados

O uso de multiplexação em duas camadas chamado de *Deeply Decoupled Parallel Processing* (D2P2) [Taifi, Khreishah and Shi 2011] permite alto desempenho e disponibilidade para sistemas que usam processadores *multicore* e GPUs, utilizando biblioteca CUDA/CUBLAS. Nesta abordagem, todos os nodos de processamento

independente (CPU + GPU) estão interconectados por um conjunto de *switch* formando um anel lógico. Uma aplicação central faz as o controle da comunicação e configuração dos nodos, existe um nodo *master* que contém o programa principal e a sequência de execução e é responsável pela distribuição das tarefas e organização dos resultados dos outros nodos, que executam a computação e enviam os resultados ao nodo *master*. Cada nodo procura um *kernel* GPU disponível para realizar a computação. Se este não for encontrado, a computação é realizada pela GPU. Em média, o aumento de desempenho é de 29%. No entanto esse valor pode variar conforme a granularidade da computação executada.

Acosta *et. al* (2011) propõe uma biblioteca de balanceamento dinâmico de carga que permite que o código paralelo seja adaptado para sistemas heterogêneos utilizando alocação de recursos CUDA. O algoritmo visa evitar desequilíbrios de carga entre as *threads* da GPU. Para resolver as diferenças de tempo na execução do código paralelo, é utilizado um esquema iterativo, onde uma operação de cálculo é realizada para cada iteração. Cada processador irá executar cálculos de acordo com o tamanho da tarefa alocada. Após, uma operação de comunicação coletiva é realizada onde todos os processadores podem sincronizar os dados antes de prosseguir para a próxima iteração. Cada processador pode saber em tempo de execução quanto tempo será utilizado para executar a tarefa atribuída. O balanceamento de carga é obtido comparando o tempo de execução real de tarefas para cada processador com a redistribuição de tarefas subsequentes.

O método *fork-join search*, consiste em dividir uma tarefa principal em sub-tarefas, processá-las individualmente e sincronizar os resultados [Ou, Chen and Lai 2011]. O ambiente consiste em nodos heterogêneos que utilizam CPUs e GPUs. Existem três tipos de nodos: Gerenciador, responsável pelo andamento da tarefa e por enviar comandos de controle ao escalonador; Escalonador, responsável por controlar as filas e distribuir as sub-tarefas aos nodos de trabalho que executam a computação. As GPUs são usadas para o processamento de alto desempenho e os CPUs são usados para receber a tarefa e enviar o resultado. As sub-tarefas são alocadas para os nodos de maneira dinâmica e as execuções são controladas por ciclos, em caso de *timeout* a sub-tarefa é considerada perdida e é realocada. Os resultados mostraram um aproveitamento de 98% da capacidade de processamento utilizando quatro nodos.

Considerando o consumo energético, Wang and Ren (2010) propõem um algoritmo de distribuição de trabalho em sistemas heterogêneos entre CPUs e GPUs. O método de balanceamento dinâmico é escalável baseado na frequência do processador visando minimizar o consumo de energia. Os resultados mostram que a distribuição de trabalho entre CPU e GPU, foi obtida redução de 14% no consumo de energia comparados com mapeamentos estáticos típicos.

A partir dos trabalhos relacionados serão necessárias pesquisas sobre a distribuição de tarefas de maneira dinâmica entre CPUs e GPUs no mesmo nodo, considerando o gargalo de transmissão PCIe e o tipo de operação a ser processada, visando obter melhor aproveitamento da arquitetura e redução no consumo de energia. Muitos dos trabalhos apresentam basicamente uma distribuição mestre-escravo, o que resulta em grande quantidade de troca de mensagens entre a CPU e GPU o que pode congestionar o barramento, comprometendo a performance da arquitetura.

Referências

"Top500 list", http://www.top500.org/lists/2011/11, Nov 2011.

Acosta, Alejandro; Corujo, Robert; Blanco, Vicente; Almeida, Francisco. (2011) "Dynamic Load Balancing on Heterogeneous Multicore/MultiGPU Systems", IEEE Computer Society, pages 467-476.

Daga, Mayank; Aji, Ashwin M.; Feng, Wu-chun. (2011) "On the Efficacy of a Fused CPU+GPU Processor (or APU) for Parallel Computing", IEEE Computer Society, pages 141-149.

Doerksen, Matthew; Solomon, Steven; Thulasiraman, Parimala. (2011) "Designing APU Oriented Scientific Computing Applications in OpenCL", IEEE Computer Society, pages 587-592.

Grimshaw, Adrew S.; Weissman, Jon B.; West, Emily A.; Loyot, Jr. (1994) "Metasystems: An approach combining parallel processing and heterogeneous distributed computing systems", Journal of Parallel and Distributed Computing, v. 21, pages 257-270.

Ou, Yanlin; Chen, Hu; Lai, Lushuang. (2011) "A Dynamic Load Balance on GPU Cluster for Fork-Join Search" , IEEE Computer Society, pages 592-596.

Taifi, Moussa; Khreishah, Abdallah; Shi, Justin Y. (2011) "Natural HPC Substrate: Exploitation of Mixed Multicore CPU and GPUs", IEEE Computer Society, pages 33-40.

Maheshwar, Piyush. (1996) "A Dynamic Load Balancing Algorithm for a heterogeneous Computing Environment", IEEE Computer Society, v. 1, pages 338-346.

Wang, Guibin and Ren, Xiaoguang (2010) "Power-efficientWork Distribution Method for CPU-GPU Heterogeneous System", IEEE Computer Society, pages 122-129.

Wang, Lei; Huang, Yong-zhong; Chen, Xin; Zhang, Chun-yan. (2008) "Task Scheduling of Parallel Processing in CPU-GPU Collaborative Environment", IEEE Computer Society, pages 228-232.