Avaliação da Auto-Adaptação de Micro-Lote para aplicação de Processamento de Streaming em GPUs

Ricardo Leonarczyk, Dalvan Griebler

[1] Escola Politécnica, Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brasil

ricardo.leonarczyk@edu.pucrs.br,dalvan.griebler@pucrs.br

Resumo. *Este artigo apresenta uma avaliação de algoritmos para regular a latência através da auto-adaptação de micro-lote em sistemas de processamento de streaming acelerados por GPU. Os resultados demonstraram que o algoritmo com o fator de adaptação fixo conseguiu ficar por mais tempo na região de latência especificada para a aplicação.*

1. Introdução

Os sistemas de processamento de streaming (SPSs) são tradicionalmente encontrados em clusters de máquinas comuns, especializando-se em prover escalabilidade horizontal. Nesses cenários normalmente o foco maior recai sobre o I/O, enquanto os requerimentos de computação não costumam ser muito altos, envolvendo lógica para realizar filtragens e transformações de dados. No entanto, quando se trata de aplicações com alta demanda computacional, como para as áreas de visão computacional, processamento de linguagem natural e robótica, a computação torna-se essencial para que se mantenha um nível de serviço aceitável. Nesses cenários, aceleradores tais como GPUs mostram-se vantajosos, dada a sua capacidade de paralelismo massivo de dados. A introdução de uma GPU em um SPS pode apresentar alguns desafios, pois processar itens pequenos pode acarretar na subutilização da GPU e na perda da vantagem sobre a CPU. Uma possível solução é processar os itens em lotes (ou micro-lotes). Contudo, constata-se que o aumento do tamanho do lote causa o aumento da latência, assim surgindo um compromisso entre latência e taxa de transferência (throughput) que pode ser regulado através da obtenção de um tamanho de lote capaz de satisfazer os requisitos de QoS da aplicação. Porém, encontrar um tamanho de lote adequado é mais um desafio, visto que as aplicações de streaming geralmente estão sujeitas a flutuações na taxa de entrada e na carga de trabalho. Isso faz com que o tamanho ideal seja um alvo móvel.

Uma possível resposta ao desafio de encontrar um tamanho de lote satisfatório é o uso de abordagens de auto-adaptação capazes de regular o referido parâmetro de acordo com informações relevantes coletadas sobre a aplicação de streaming durante sua execução. A abordagem mais comum nesse sentido é o uso de algoritmos de decisão on-line, que consideram medições anteriores de parâmetros como latência ou taxa de transferência para adaptar o tamanho do lote e/ou o grau de paralelismo. O trabalho de [De Matteis et al. 2019] apresenta dois algoritmos capazes de adaptar o tamanho de lote para minimizar a latência e maximizar a taxa de transferência, porém assume-se uma carga de trabalho estável. Já o trabalho de [Stein et al. 2021] apresenta quatro algoritmos de adaptação que suportam cargas de trabalho dinâmicas e que mostraram-se satisfatórios para o controle da latência da aplicação de compressão LZSS Os algoritmos de adaptação propostos têm como parâmetro a latência desejada, um limiar de tolerância (em porcentagem) em torno da latência desejada, e um fator de adaptação (tamanho do passo para

obter um novo tamanho de lote). Diante disso, o objetivo deste estudo é avaliar os algoritmos propostos por [Stein et al. 2021], para entender se o comportamento observado na aplicação LZSS pode ser generalizado para outro tipo de aplicação de streaming em GPU.

A aplicação escolhida para implementação dos algoritmos no presente estudo chama-se Militar Server (MS). Trata-se de uma aplicação de streaming sintética projetada para explorar o paralelismo de dados. Possui em sua pipeline (de 5 estágios) padrões paralelos de mapeamento e redução sobre estruturas como matrizes. Foi implementada com as bibliotecas Fastflow e GSParLib. O problema de domínio da aplicação é fazer a alocação de unidades militares em um mapa, levando-se em conta o seu tipo e requerimentos de localização. A computação para a alocação é feita por um servidor, o qual continuamente recebe dados enviados por drones que sobrevoam as regiões do mapa. Em comparação com a LZSS, a MS possui dois estágios a mais, e seus itens consistem em estruturas (*structs*) que não podem ser quebradas, enquanto no LZSS os itens são fragmentos de arquivos (expresso em *bytes*).

2. Resultados e Conclusões

Os experimentos foram realizados em uma máquina com memória de 32GB, processador AMD Ryzen 5 de 12 núcleos lógicos, e uma GPU NVIDIA GeForce RTX 3090 com 24GB de memória e 10496 núcleos CUDA. A carga de trabalho da aplicação consistiu em 500.000 itens, mantendo-se a taxa de entrada estável, porém com requerimentos de computação variáveis. A latência alvo foi de 3ms, com limiares especificados na tabela 1. A métrica usada para comparação foi o número de acertos de SLO (objetivo de nível de serviço), que é uma porcentagem calculada através da divisão do número de lotes que ficaram dentro do limiar de tolerância pelo número total de lotes emitidos pela aplicação. Utilizou-se a média de 10 execuções para cada combinação de parâmetros.

Tabela 1. Resultados dos acertos de SLO (em porcentagem) por algoritmo (linhas) e por limiar (colunas). Todos os valores são porcentagens.

Algoritmo	5%	10%	15%	20%	50%
Fator Fixo (FAF)	23,21	38,12	48,47	56,66	79,77
Fator Percentual (PBAF)	19,97	30,7	38,51	47,49	77,87
Fator Percentual Sem Limiar (PBAF Without Trheshold)	19.47	30.44	38.73	47.20	78.19
Fator Múltiplo (MBAF)	19,48	31,61	40,96	48,92	70,36

No geral, o algoritmo que mantém o fator de adaptação fixo (denominado FAF) obteve mais acertos de SLO. Os demais algoritmos, os quais são capazes de ajustar o fator de adaptação de acordo com a proximidade do limiar, obtiveram uma redução de cerca de 16% na referida métrica. A penalidade ocorre devido à necessidade de fazer grandes ajustes no tamanho do lote para manter a latência alvo em uma carga de trabalho altamente dinâmica. Reduzir o fator de adaptação não se mostrou eficaz neste caso.

Referências

De Matteis, T., Mencagli, G., De Sensi, D., Torquati, M., and Danelutto, M. (2019). Gasser: An auto-tunable system for general sliding-window streaming operators on gpus. *IEEE Access*, 7:48753–48769.

Stein, C. M., Rockenbach, D. A., Griebler, D., Torquati, M., Mencagli, G., Danelutto, M., and Fernandes, L. G. (2021). Latency-aware adaptive micro-batching techniques for streamed data compression on graphics processing units. volume 33, page e5786.