Paralelização de Cálculos Estatísticos sobre Dados de Monitoramento da Camada de Ozônio: um Estudo com GPU

Lucas L. Nesi[1], **Guilherme P. Koslovski**[1], **Andrea S. Charão**[2]
Damaris K. Pinheiro[2], **Luiz Angelo Steffenel**[3]

[1]Universidade do Estado de Santa Catarina (UDESC)

[2]Universidade Federal de Santa Maria (UFSM)

[3]Université de Reims Champagne-Ardenne – França

Resumo. *O presente trabalho descreve a paralelização, em GPU, de cálculos com dados de satélites que monitoram a camada de ozônio, visando apoiar investigações sobre sua dinâmica. A paralelização organiza as operações em pipeline e considera a arquitetura de memória da GPU. Utilizando dados do período de 2005 a 2016, apresenta-se resultados de desempenho usando diferentes arquiteturas de processamento, com armazenamento em HDD e SSD.*

1. Introdução

A redução da camada de ozônio terrestre é alvo de muitos estudos, pois seu efeito é nocivo ao meio ambiente e à saúde da população mundial. Pesquisadores têm à disposição muitos dados públicos de monitoramento por instrumentos, mas ainda há poucos estudos sobre modelagem numérica da dinâmica circulatória da camada de ozônio, que poderiam ajudar em detecções e predições tanto de fenômenos transitórios como de longo prazo [Steffenel et al. 2016]. Esse tipo de estudo requer técnicas de processamento de alto desempenho que agilizem a análise de grandes quantidades de dados.

A utilização de unidades de processamento gráfico (GPU, do inglês *Graphics Processing Unit*) é uma alternativa de alto desempenho e de baixo custo para uma vasta gama de aplicações [Plauth et al. 2016]. A elevada capacidade de processamento de GPUs (superando 1 TFlops), em conjunto com a consolidação das ferramentas para programação com propósito geral, CUDA (Compute Unified Device Architecture) em particular, servem de motivação para este trabalho, cuja abordagem difere-se de estudos precedentes [Steffenel et al. 2016].

Neste trabalho, apresenta-se a paralelização de cálculos estatísticos sobre uma base de dados pública, resultante do monitoramento da camada de ozônio por satélite. O restante deste texto descreve os dados de entrada e os cálculos estatísticos (Seção 3), a estratégia de paralelização baseada em *pipeline* (Seção 4), e os resultados de desempenho em diferentes arquiteturas com GPU, com dados armazenados em HDD e SSD (Seção 5).

2. Trabalhos Relacionados

A questão da utilização de SDD e HDD é discutida por [Polte et al. 2008]. Extensivas comparações sobre a quantidade de escritas e leituras por segundo indicaram que o SSD tem maior desempenho nestes aspectos. A utilização de GPUs para controlar um sistema de armazenamento distribuído com SSDs foi investigada por [Gharaibeh et al. 2010], acelerando a vazão final do sistema. Ainda, os autores de [Onishi et al. 2015] indicam

que a capacidade de processamento cresceu mais rapidamente que a vazão de entrada e saída, justificando a aplicação de soluções combinadas (como a *pipeline* do presente trabalho). Por fim, a utilização de *pipelines* usando GPUs para a mineração de dados em tempo real é explorada por [Edgar et al. 2010]: dados coletados por um telescópio resultaram em um fluxo de dados de $5GiB/s$, e a utilização de uma *pipeline* é essencial para manter o fluxo de processamento constante.

3. Cálculo de Estatísticas com Dados sobre a Camada de Ozônio

Os dados utilizados neste trabalho são disponibilizados pela NASA[1] e são coletados diariamente por equipamentos a bordo de um satélite. Tais dados, disponibilizados desde novembro de 2005, representam medições da espessura da camada de ozônio em um único dia, sobre coordenadas geográficas fixadas em uma área variando em $1,00$ grau latitudinal e $1,00$ graus longitudinal, dispostas em uma matriz com 180 linhas e 360 colunas. Os arquivos são disponibilizados no formato CDTOMS L3 [McPeters et al. 1998].

Os cálculos estatísticos obtêm o mínimo, máximo, média, variância, desvio padrão, obliquidade e curtose das medições da camada de ozônio, para um conjunto de coordenadas geográficas, num dado período temporal. O conjunto de coordenadas geográficas pode ser definido por uma linha (Latitude), uma coluna (Longitude), ou uma célula (Latitude e Longitude). O período temporal representa o intervalo de datas dos dados: quanto maior o período, mais arquivos são processados. As três primeiras métricas (mínimo, máximo e média) são facilmente calculadas na operação de união de dois elementos. Para os cálculos de variância, desvio padrão, obliquidade e curtose, são usados os acúmulos dos momentos centrais de segundo, terceiro e quarto grau.

4. Paralelização com GPU

Para aproveitar todos os recursos do sistema em qualquer instante de execução, o processo é organizado em forma de *pipeline*. Assim, quando um arquivo tem sua leitura finalizada, pode-se processá-lo na GPU sem interromper a leitura do próximo arquivo. A *pipeline* é dividida em três módulos principais: Leitores, *CudaWorkers* e Escritores. O número de elementos em cada módulo pode variar a fim de explorar melhor os recursos do sistema.

Os Leitores são encarregados das leituras dos arquivos no formato original. Três estratégias foram utilizadas para acelerar a leitura: *(i)* utilização de diferentes quantidades de módulos (*Threads*) leitoras; *(ii)* conversão do formato textual para binário; e *(iii)* utilização de diferentes mídias físicas de armazenamento. Os *CudaWorkers* são responsáveis pelas cópias dos dados para a memória da GPU e pela execução dos *kernels* CUDA. Ainda, adotou-se uma estratégia para usar a memória compartilhada da GPU: cada arquivo teve seus dados particionados em submatrizes e cada submatriz foi alocada para um bloco CUDA, levando em consideração a capacidade da memória compartilhada.

Cada submatriz tem suas estatísticas calculadas para cada linha e coluna. Inicialmente, cada submatriz é copiada para memória compartilhada e são realizados os cálculos estatísticos. Cada célula da matriz seria lida duas vezes em memória global, uma para o cálculo da linha e outro para o cálculo da coluna, mas com sua passagem para memória compartilhada, apenas um único acesso é necessário. O *kernel* CUDA

[1]Dados disponíveis em: http://ozoneaq.gsfc.nasa.gov/

para calcular as estatísticas é invocado com tamanho de blocos $2x4$, totalizando as 8 submatrizes e $90x90$ *threads*, uma *thread* para cada linha e coluna.

Após este processo, existe uma redução de todas as 8 subestatísticas calculadas, resultando nas estatísticas da matriz como um todo. O cálculo da redução é basicamente a união das submatrizes de forma coerente. Durante o processo, as células são individualmente unificadas com o acumulador global de cada *CudaWorker*. Este acumulador global contém as uniões de todas as estatísticas das células, linhas e colunas de todos os arquivos processados neste *CudaWorker* até o momento. No fim de todas as operações, uma redução entre os acumuladores globais de cada *CudaWorker* resulta nas estatísticas de todos os arquivos.

5. Análise Experimental

Realizou-se experimentos para comparar os tempos de processamento entre os módulos da *pipeline* em diferentes arquiteturas de processamento e leitura. Os testes foram realizados com 3915 arquivos, correspondentes aos dados diários de $11/2005$ até $10/2016$. Foram usados 3 tipos diferentes de arquiteturas que são descritas na Tabela 1.

Tabela 1. Configuração dos hardwares de teste.

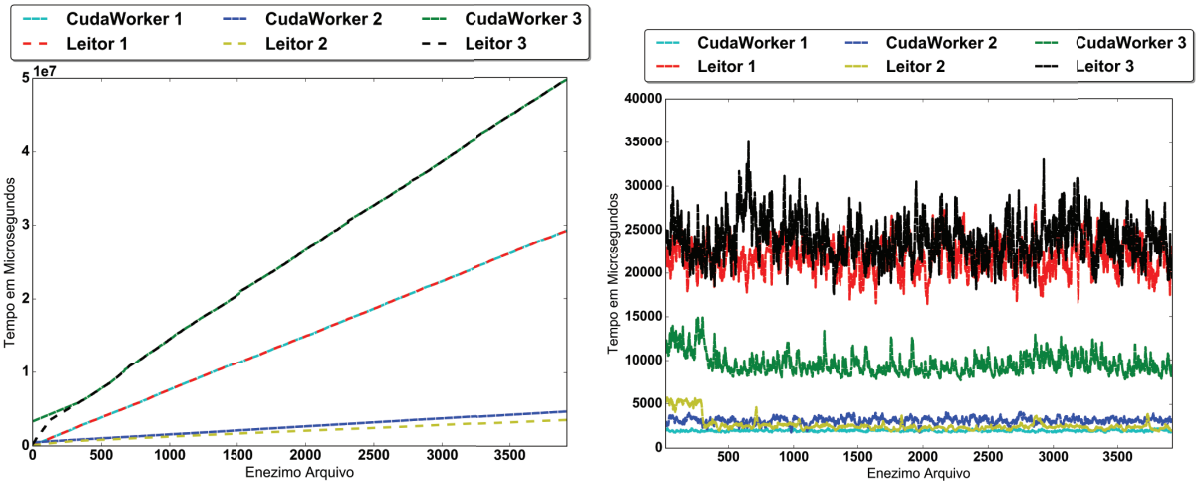
Conf.	Processador	Memória	GPU	Armazenamento
1	Intel i7 2600K 3.4GHz	32GB 1600MHz	Geforce GTX 1080	HDD 7200 RPM
2	Intel i7 2600K 3.4GHz	32GB 1600MHz	Geforce GTX 1080	SSD 6 Gbps
3	Intel i7 4770 3.40GHz	16GB 1600MHz	Geforce GTX 730	HDD 7200 RPM

A Figura 1(a) mostra o tempo acumulado para ler e processar o enésimo arquivo, isto é, o tempo de execução desde quando o arquivo foi efetivamente lido, e o tempo no qual ele foi processado. A Figura 1(b) mostra a média móvel dos tempos individuais de leitura e processamento de cada arquivo. Pode-se estabelecer uma comparação entre a vazão dos módulos de leitura e *CudaWorker*, e entre as diferentes arquiteturas. Observa-se na Figura 1(b) que a leitura usando HDD é superior ao tempo de processamento em qualquer situação, causando na Figura 1(a) a sobreposição das linhas de processamento das configurações 1 e 3 em relação as linhas de leitura. Porém, a Figura 1(b) mostra que o desempenho da leitura usando SSD é superior ao de processamento do mesmo arquivo. Assim, na Figura 1(a), o tempo de processamento do último arquivo, por exemplo, é superior ao seu tempo de leitura. Ou seja, existe uma sobrecarga causada pelo processamento dos arquivos.

Desta forma, nota-se que usar *hardware* de leitura de baixo desempenho (HDD) resulta no acúmulo de trabalho nos leitores, já que o tempo de uma leitura é superior ao tempo do processamento daqueles dados na GPU. Assim, o ganho de desempenho na GPU é limitado ao tempo de leitura do arquivo. Com a utilização de SSD, o gargalo da *pipeline* é o processamento dos dados, fato que só poderia ser melhorado revendo-se as técnicas usadas por este processamento, ou utilizando-se mais de uma GPU nesta parte da *pipeline*. Ressalta-se a importância da *pipeline* para garantir o uso mútuo e contínuo das funções de entrada e saída e de GPU.

6. Conclusão

Este artigo apresentou a utilização de uma *pipeline* de processamento utilizando GPU com CUDA para o processamento de dados sobre a camada de ozônio. Tendo em



(a) Tempos acumulados.

(b) Média móvel dos tempos individuais.

Figura 1. Gráficos comparando o tempo entre os módulos da *pipeline* Leitor e *CudaWorker* nas diversas configurações de *hardware* propostas.

vista os resultados observados em diferentes tipos de arquitetura e sutis mudanças na *pipeline*, fica evidenciado a necessidade da utilização de *hardware* de armazenamento com grandes taxas de leitura. No caso apresentado, a utilização de um SSD foi suficiente para transferir o gargalo do sistema para a GPU. A partir disso, trabalhos futuros podem explorar melhores formas de leitura com diferentes tipos de *hardware*, o particionamento dos dados na GPU com uma melhor representação, ou melhores otimizações durante a realização dos cálculos e das reduções.

Referências

Edgar, R., Clark, M., Dale, K., Mitchell, D., Ord, S., Wayth, R., Pfister, H., and Greenhill, L. (2010). Enabling a high throughput real time data pipeline for a large radio telescope array with GPUs. *Computer Physics Communications*, 181(10):1707 – 1714.

Gharaibeh, A., Al-Kiswany, S., Gopalakrishnan, S., and Ripeanu, M. (2010). A GPU accelerated storage system. In *Proc. of the Int. Symp. on High Performance Distributed Computing*, pages 167–178. ACM.

McPeters, R., Bhartia, P., Krueger, A., and Herman, J. (1998). *Earth Probe Total Ozone Mapping Spectrometer (TOMS) Data Products User's Guide*. NASA.

Onishi, S., Chakarothai, J., Ishii, T., Hashikawa, S., and Suzuki, Y. (2015). Acceleration of I/O data transfer with RDMA for massively large-scale GPU simulation. In *GPU Technology Conference*.

Plauth, M., Feinbube, F., Schlegel, F., and Polze, A. (2016). A performance evaluation of dynamic parallelism for fine-grained, irregular workloads. *Int. Journal of Networking and Computing*, 6(2):212–229.

Polte, M., Simsa, J., and Gibson, G. (2008). Comparing performance of solid state devices and mechanical disks. In *3rd Petascale Data Storage Workshop*, pages 1–7.

Steffenel, L. A., Pinheiro, M. K., Pinheiro, D. K., and Perez, L. V. (2016). Using a pervasive computing environment to identify secondary effects of the antarctic ozone hole. *Procedia Computer Science*, 83:1007 – 1012.