Trabalho Final

Computação Paralela com GPU

Brendon Henrique
bhps17@inf.ufpr.br

GRR20170203

Universidade Federal do Paraná
Bacharelado em Ciência da Computação

Análise de desempenho do algoritmo de ordenação bitonic usando CUDA

1 Descrição do problema:

O objetivo do trabalho proposto é analisar o desempenho/performance do algoritmo bitonic sort que se trata de um algoritmo paralelo clássico para ordenação em relação a algoritmos sequenciais. O algoritmo será implementado em CUDA para executar de forma paralela em GPU.

2 Como executar:

Para executar o bitonic sort em gpu, primeiro é necessário compilar o arquivo "bitonic_sort.cu". Ao executar o arquivo binário indicar algum tamanho do vetor válido através da flag "-n". Caso seja desejado imprimir os vetores gerados pré/pós ordenação usar a flag "-p" (opcional). Os valores de entrada validos estão no próximo tópico do relatório. Para a execução da versão sequencial deve-se seguir os mesmos passos descritos anteriormente.

3 Descrição da entrada:

Como dados de entrada para ordenação serão gerados números pseudo-aleatórios para preenchimento dos vetores a serem ordenados, números gerados serão floats. Serão gerados e preenchidos vetores de tamanhos 1024, 8192, 131072, 1048576, 16777216, de forma a alocar os vetores gerados de forma performática nas threads e blocos da versão paralela.

4 Parametrização da versão paralela:

Tam vetor	2^Exp	Threads	Blocos
1024	2^9	4	256
8192	2^13	8	1024
131072	2^17	32	4096
1048576	2^21	128	16384
16777216	2^24	512	32768

Table 1: Setup para melhor adequação de threads e blocos de acordo com tamanho do vetor de entrada

5 Configuração:

As métricas(tempos de execução) descritas a seguir foram coletadas em um computador com a seguinte configuração: Processador Intel i5-7600k 3.80 GHz, GTX 1060 6GB, Memória 16 GB.

6 Descrição das métricas de medição:

Para agilizar o desenvolvimento e focar somente na implementação da versão paralela em GPU do algoritmo foi obtido do site (https://www.geeksforgeeks.org/bitonic-sort/) a versão sequencial do algoritmo.
O desempenho do algoritmo da versão paralela também foi comparado com o algoritmo de ordenação padrão do C++, o IntroSort para uma execução sequencial.

Para obtenção dos dados gerados foram executadas 10 execuções dos algoritmos e retirado o melhor desempenho entre essas execuções para representar o desempenho do algoritmo. Para obtenção dos tempos foi utilizado a biblioteca "chono" disponibilizada pelo professor, com pequenas modificações na função que realizava o cálculo da diferença de tempo entre as marcações para medição. As medições obtidas dizem respeito somente a parte onde a ordenação era efetuada em ambos algoritmos.

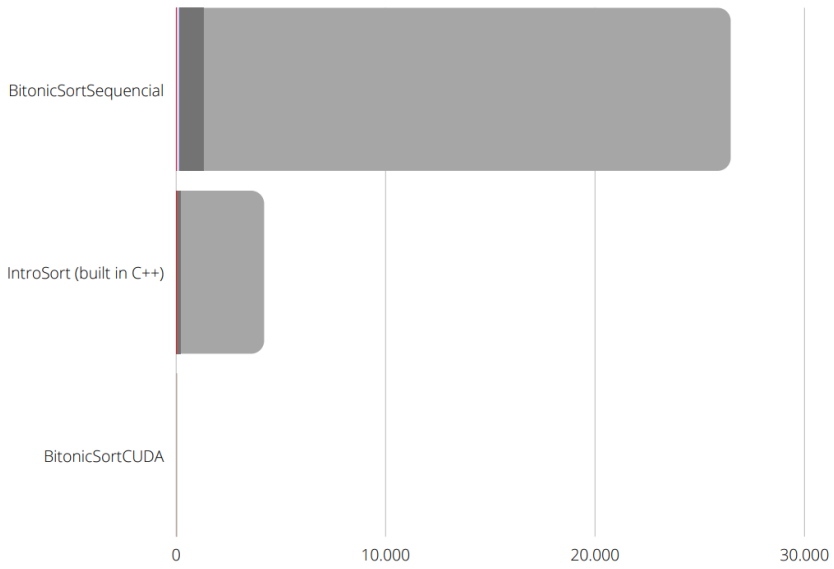

Figure 1: Tempos por execução em milissegundos dos algoritmos

Como podemos ver nos gráficos obtidos das execuções dos algoritmos podemos ver o quanto o bitnonic sort em CUDA é superior em relação a performance comparado as outras abordagens. O desempenho é tão destoante que não é possível distinguir as barras representando as execuções do bitonic sort paralela, pois os tempos coletados são muito pequenos.

Para conseguir realizar a comparação de via gráfico de uma forma que seja possível visualizar os tempos de execução do bitonic sort em cuda foi necessário remover as medições com vetores maiores, pois com vetores maiores já como o esperado o bitonic sort paralelo é muito superior em desempenho ficando impossível comparar. Assim como também foi necessário remover da comparação a versão do bitonic sort sequencial, pois além do mesmo utilizar uma implementação recursiva, o que degrada seu desempenho ele é um algoritmo conhecidamente bom para a utilização de forma paralela.

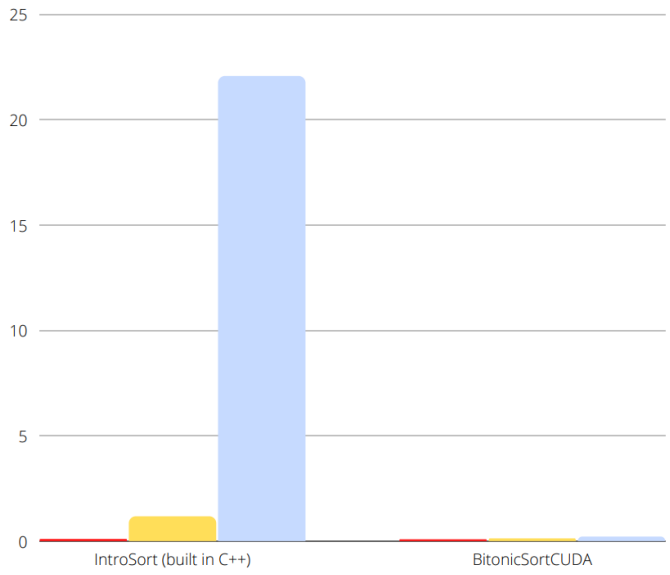

Figure 2: Tempos por execução em milissegundos dos algoritmos

Fica evidente o quanto o bitonic sort performa bem com vetores maiores, não aumentando em grande magnitude o tempo que o algoritmo leva para resolver a ordenação, algo que acontece em outras implementações.

Os tempos utilizados estão em milissegundos e podem ser vistos com maior precisão na tabela a seguir:

	BitonicSortCUDA	IntroSort (built in C++)	BitonicSortSequencial
1024	0.077648	0.101744000	0.874694000
8192	0.116965000	1.176724000	15.567610000
131072	0.192569000	22.050786000	131.248249000
1048576	0.308517000	207.233631000	1160.036940000
16777216	0.348511000	3925.389738000	25124.251757000

Table 2: Tempos por execução de cada algoritmo em ms

7 Conclusão:

É possível concluir com a execução deste trabalho o quanto o paralelismo a nível de GPU pode agregar ao desempenho desde a realização de tarefas simples, como ordenação por exemplo como tarefas mais complexas. Mesmo sendo um pouco desleal a comparação de uma versão paralela de um algoritmo com a versão sequencial, não era esperado que o gap de desempenho entre versões seria tão assustador, o que abre caminho para pensarmos no quanto de evolução relacionado a desempenho computacional podemos gerar com a popularização e estudo da utilização de GPUs em aplicações e problemas que hoje não fazem uso desse advento.