Assignment 3 Tutorial

MSBD5009/COMP5112 - Parallel Programming

CUDA Programming

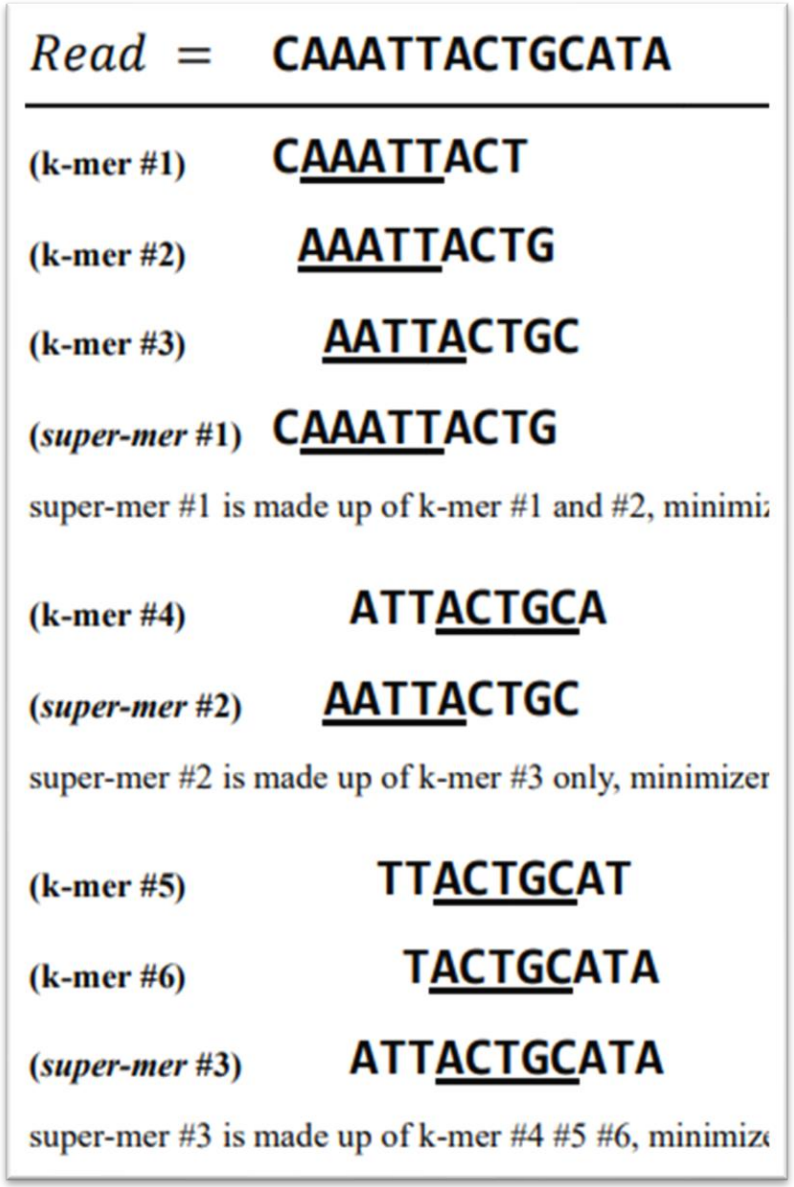
9 Apr 2022

Tutorial Overview

- Problem Description
- Task Description
- Implementation Instruction

Problem Description

- Supermer Generation
  - a **Read** has multiple **K-mers**
  - a **K-mer** has one **minimizer** (value)
  - consecutive **K-mers** with the same minimizer value will generate a **Supermer**



Task Description

- Supermer Generation with CUDA
  - Data parallelism
  - General idea - two stages:
    - 1. Generate minimizers of all k-mers in a read.
      - Read sequence: ACTGACTG …
      - Its kmers: kmer #1, kmer #2, kmer #3 …
      - Its minimizer sequence: minimizer for kmer#1, minimizer for kmer #2 …
    - 2. Find supermer beginning positions
      - A new supermer begins when the current minimizer ≠ the last minimizer
      - Find consecutive minimizers with the same value

Task Description

- The given function is **GenerateSupermer_GPU**
  - `void GenerateSupermer_GPU(vector<string> &reads, int K, int P, vector<string> &all_supermers, int NUM_BLOCKS_PER_GRID, int NUM_THREADS_PER_BLOCK)`

  - Organize read data in CSR format,
  - Malloc arrays on both host and device,
  - Memcpy the host arrays to device arrays,
  - Call the two kernel functions,
  - Memcpy device data back to host,
  - Generate and save the supermers to the vector.

Implementation Instruction

- You need to finish two kernel functions in **gensupermer_cuda.cu**:
  - GPU_GenMinimizer
    - `__global__ void GPU_GenMinimizer(T_GPU_data data, int K_kmer, int P_minimizer)`
    - Input: reads in CSR format
    - Output: minimizer sequences
  - GPU_GenSKM
    - `__global__ void GPU_GenSKM(T_GPU_data data, int K_kmer, int P_minimizer)`
    - Input: minimizer sequences
    - Output: supermer offset sequences

Implementation Instructions

- Data Structures
  - Stage 1: **GPU_GenMinimizer**
    - Input: reads in CSR format
      - ***char *reads:*** the read data
      - ***int *reads_offs:*** the read data offsets
    - Output: minimizer sequences
      - ***unsigned int *minimizers:*** the 2bits-compressed minimizers (an unsigned int sequence whose length is n-k+1. You can use the offset array ***reads_offs*** for this array)
    - 2bits-compression: 'A' -> 0b00, 'C' -> 0b01, 'G' -> 0b10, 'T' -> 0b11
      - So that a minimizer (p<=16) can be saved in a 32-bit unsigned int for better accessibility and easy comparison.
      - d_basemap is for easy type casting from char to unsigned int.
      - A sample usage:
        - `Unsigned int minimizer = 0;`
        - In a for-loop do: `minimizer = (minimizer << 2) | d_basemap[read[i]];`
        - (e.g., p=6, the minimizer AAACCT will be transformed to an integer 0b 00 00 00 01 01 11.)

Implementation Instructions

- Data Structures
  - Stage 1: **GPU_GenMinimizer**
    - Input: reads in CSR format
      - ***char *reads:*** the read data
      - ***int *reads_offs:*** the read data offsets
    - Output: minimizer sequences
      - ***unsigned int *minimizers:*** the 2bits-compressed minimizers (you can use the offset array ***reads_offs*** for this array)
    - Example (2 reads, k=8, p=4)
      - *reads: AAAACCTT CAAACCTTGG
      - *reads_offs: 0, 8, 18
      - *read_len: 8, 10
      - *minimizers: 0b00000000, -, -, -, -, -, -, -,
        0b00000001, 0b00000001, 0b00000101, -, -, -, -, -, -, -

Implementation Instructions

- Data Structures
  - Stage 2: **GPU_GenSKM**
    - Input: minimizer sequences
      - ***unsigned int *minimizers:*** the 2bits-compressed minimizers (this array shares the same offsets with *reads)
      - ***int *reads_offs:*** the read data offsets (used also as the minimizer sequence offsets)
    - Output: supermer offset sequences
      - ***int *supermer_offs:*** the beginning position in read of each supermer, the last item should be n-k+1 as the ending indicator (this array shares the same offset array ***reads_offs*** with *reads)
    - Example (2 reads, k=8, p=4)
      - `*reads:` `AAAACCTT CAAACCTTGG`
      - `*reads_offs:` `0, 8, 18`
      - `*minimizers:` **`0`**`, -, -, -, -, -, -, -, `**`1, 1, 5`**`, -, -, -, -, -, -, -`
      - `*supermer_offs:` `0, 1, -, -, -, -, -, -, 0, 2, 3, -, -, -, -, -, -, -`

      The last item in each offset sequence is $n - k + 1$ *(the underlined item).*

Thank you

MSBD5009/COMP5112 - Parallel Programming

A3: CUDA Programming