

Computer Architecture

Final Project Part 1

Accelerate RNA Sequence Quantification on RISC-V
Distributed Cluster on Near-DRAM Platform

Group:25

Member:

313512041 朱宥勳

313512035 侯書維

313512085 張嘉麟

313512071 陳柏皓

A. Distribute.cpp

To achieve workload balancing among Logic Units during DNA sequence quantification, we initially implemented a straightforward sequence distribution strategy. This baseline approach assigns sequences evenly based on their count, without considering the actual length of each sequence. While this method is effective when sequence lengths are relatively uniform, it may lead to imbalance in computational workload when sequence lengths vary significantly. Our implementation follows the same equal-count strategy used in the default code provided by the TAs, but we optimize its structure and performance by eliminating redundant operations and encapsulating the logic into a reusable function.

a. Algorithm

1. Equal Distribution Strategy

In this project, we adopted a straightforward equal distribution strategy, assigning transcript and query sequences evenly based on their count to each Logic Unit. This method does not consider sequence length, but instead aims for a balanced number of sequences across units. It is effective when sequence lengths are uniform, and provides a simple and fast baseline approach.

2. Comparison with the Default Distribution

While our approach uses the same equal-count strategy as the default distribution function (`distribute_default()`), we improved upon it by refactoring the logic into a reusable lambda function and eliminating redundant initialization and copy operations. This significantly increases performance, especially when handling large datasets. As a result, our implementation maintains the simplicity of the original method but achieves faster execution and cleaner structure.

b. Performance Result

1. Modified version

```
root@e1a4699c286c:/home/CA-FP1/evaluation# make run_distribute
./distribute ../data/gencode.v43.transcripts.noN.fa ../data/queries.10K.fa 15 256 1
Passed
Distribution took 433.85 milliseconds.
225327 121.515
1.70975e+06 98046.9
Writing file transcripts_data.h
✓ Wrote to transcripts_data.h successfully.
Writing file query_data.h
✓ Wrote to query_data.h successfully.
root@e1a4699c286c:/home/CA-FP1/evaluation#
```

2. Default version

```
root@e1a4699c286c:/home/CA-FP1/evaluation# make run_distribute
./distribute ../data/gencode.v43.transcripts.noN.fa ../data/queries.10K.fa 15 256 1
Passed
Distribution took 725.968 milliseconds.
225327 121.515
1.70975e+06 98046.9
Writing file transcripts_data.h
✓ Wrote to transcripts_data.h successfully.
Writing file query_data.h
✓ Wrote to query_data.h successfully.
root@e1a4699c286c:/home/CA-FP1/evaluation# vi distribute.cpp
```

B. Index.c

The goal of this stage is to generate all possible k-mers of length k from multiple DNA/RNA (or arbitrary character) sequences and insert each k-mer into a hash map (HashMap). To achieve this efficiently, we design an algorithm that uses the Rolling Hash technique to significantly reduce redundant computations.

To improve the performance and scalability of the hash map implementation, we made several key optimizations to the original design. First, we replaced the basic DJB2 hash function with a custom hash function that operates on the first 15 characters using bitwise operations. This reduces collision probability for short fixed-length keys and accelerates hashing. Second, instead of incrementally increasing capacity by a fixed value, we now double the capacity when resizing the hashmap. This reduces the number of reallocations and improves amortized insertion time. Additionally, during resizing, we reinsert each value individually using the public insert function, which ensures correct hash computation and value handling, especially under chaining collisions. Finally, we optimized key comparison logic by replacing `strcmp()` with `strncmp()` to enhance performance in scenarios involving prefix-matching or fixed-length keys.

a. Algorithm

1. Rolling Hash

The algorithm first computes the hash value of the first k-mer in each sequence. Then, as it slides forward by one character, it only needs to remove the previous character and add the new character to quickly compute the hash value of the next k-mer in $O(1)$ time. Although each k-mer still needs to be copied as a string and inserted into the HashMap, the overall efficiency of hash value computation is greatly improved.

b. Performance Result

1. Modified version

```
root@eia4699c286c:/home/CA-FPI/evaluation# make run_index
/home/gen5/build/RISCV/gen5.opt /home/gen5/configs/example/se.py -c index.o -o "15" --cpu-type=MnlorCPU --cpu-clock=800MHz --mem-type=Ramulator --ramulator-config=/home/gen5/configs/ramulator/LPDDR4-config.cfg --mem-size=256MB --caches --l1i_size=32kB --l1i_assoc=8 --l1d_size=32kB --l1d_assoc=8 --cacheline=32
gen5 Simulator System. http://gen5.org
gen5 is copyrighted software; use the --copyright option for details.

gen5 version 20.1.0.0
gen5 compiled Feb 7 2022 09:12:04
gen5 started May 20 2025 13:00:58
gen5 executing on eia4699c286c, pid 2775
command line: /home/gen5/build/RISCV/gen5.opt /home/gen5/configs/example/se.py -c index.o -o 15 --cpu-type=MnlorCPU --cpu-clock=800MHz --mem-type=Ramulator --ramulator-config=/home/gen5/configs/ramulator/LPDDR4-config.cfg --mem-size=256MB --caches --l1i_size=32kB --l1i_assoc=8 --l1d_size=32kB --l1d_assoc=8 --cacheline=32
warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
Ramulator system configuration file = /home/gen5/configs/ramulator/LPDDR4-config.cfg
MEMORY TYPE:: Ramulator
Ramulator
warn: membus.master is deprecated. 'master' is now called 'mem_side_ports'
Global frequency set at 100000000000 ticks per second
using riscv64 ISA
warn: Unknown operating system; assuming Linux.
0: system.remote.gdb: listening for remote gdb on port 7000
using LPDDR4X 4266MHz
**** REAL SIMULATION ****
info: Entering event queue @ 0. Starting simulation...
reset tick: 14287500
dump tick: 67567382500
Simulated Time : 0.067553 seconds
Exiting @ tick 80536253750 because exiting with last active thread context
rm -rf index_result/nsout
mv nsout index_result/
root@eia4699c286c:/home/CA-FPI/evaluation# vi distribute.cpp
```

2. Default version

```
root@eia4699c286c:/home/CA-FPI/evaluation# make run_index
/home/gen5/build/RISCV/gen5.opt /home/gen5/configs/example/se.py -c index.o -o "15" --cpu-type=MnlorCPU --cpu-clock=800MHz --mem-type=Ramulator --ramulator-config=/home/gen5/configs/ramulator/LPDDR4-config.cfg --mem-size=256MB --caches --l1i_size=32kB --l1i_assoc=8 --l1d_size=32kB --l1d_assoc=8 --cacheline=32
gen5 Simulator System. http://gen5.org
gen5 is copyrighted software; use the --copyright option for details.

gen5 version 20.1.0.0
gen5 compiled Feb 7 2022 09:12:04
gen5 started May 20 2025 05:01:24
gen5 executing on eia4699c286c, pid 2190
command line: /home/gen5/build/RISCV/gen5.opt /home/gen5/configs/example/se.py -c index.o -o 15 --cpu-type=MnlorCPU --cpu-clock=800MHz --mem-type=Ramulator --ramulator-config=/home/gen5/configs/ramulator/LPDDR4-config.cfg --mem-size=256MB --caches --l1i_size=32kB --l1i_assoc=8 --l1d_size=32kB --l1d_assoc=8 --cacheline=32
warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
Ramulator system configuration file = /home/gen5/configs/ramulator/LPDDR4-config.cfg
MEMORY TYPE:: Ramulator
Ramulator
warn: membus.master is deprecated. 'master' is now called 'mem_side_ports'
Global frequency set at 100000000000 ticks per second
using riscv64 ISA
warn: Unknown operating system; assuming Linux.
0: system.remote.gdb: listening for remote gdb on port 7003
using LPDDR4X 4266MHz
**** REAL SIMULATION ****
info: Entering event queue @ 0. Starting simulation...
reset tick: 14175000
dump tick: 2545945005000
Simulated Time : 2.545931 seconds
Exiting @ tick 2556048171250 because exiting with last active thread context
rm -rf index_result/nsout
mv nsout index_result/
root@eia4699c286c:/home/CA-FPI/evaluation# vi index.c
```

3. Evaluation Performance

	Index default	Index modify
10 sequences with 5953 characters	3.042921s	0.067553s
Full with 2703168 characters	1381.74s	30.664s

C. Quantify.c

To efficiently perform the query phase of RNA sequence quantification, we implemented an optimized `quantify()` function that processes all k-mers in each query sequence to identify the most frequently matched transcript sequences. By introducing a sentinel round marking technique, the algorithm avoids resetting large tracking arrays between queries, significantly reducing memory operations and conditional branches.

a. Algorithm

1. Sentinel Round Optimization

This function applies a “sentinel round” concept, where a round counter represents the current query index. Combined with the `last_updated` array, this eliminates the need to clear the entire counts array after every query, making the process more efficient.

2. Max Voting on Transcript Matches

Since each query can match multiple transcripts, the algorithm records the transcript(s) with the highest match count for that query and increments their count in the `final_results`. This simulates a max-voting approach to determine the most likely transcript origin for each query sequence.

b. Performance Result

1. Modified version

```
root@e1a4699c286c:/home/CA-FP1_quantfly_new/evaluation# make run_quantify
/home/gen5/build/RISCV/gen5.opt /home/gen5/configs/example/se.py -c quantify.o -o "15" --cpu-type=MnorrCPU --cpu-clock=800MHz --mem-type=Ranulator --ranulator-config=/home/gen5/configs/ranulator/LPDDR4-config.cfg --mem-size=256MB --caches --l1l_size=32kB --l1l_assoc=8 --l1d_size=32kB --l1d_assoc=8 --cacheline=32
gen5 Simulator System. http://gen5.org
gen5 is copyrighted software; use the --copyright option for details.

gen5 version 20.1.0.0
gen5 compiled Feb 7 2022 09:12:04
gen5 started May 20 2025 13:02:15
gen5 executing on e1a4699c286c, pid 2788
command line: /home/gen5/build/RISCV/gen5.opt /home/gen5/configs/example/se.py -c quantify.o -o 15 --cpu-type=MnorrCPU --cpu-clock=800MHz --mem-type=Ranulator --ranulator-config=/home/gen5/configs/ranulator/LPDDR4-config.cfg --mem-size=256MB --caches --l1l_size=32kB --l1l_assoc=8 --l1d_size=32kB --l1d_assoc=8 --cacheline=32

warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
Ranulator system configuration file = /home/gen5/configs/ranulator/LPDDR4-config.cfg
MEMORY TYPE: Ranulator
Ranulator
warn: membus.master is deprecated. 'master' is now called 'mem_side_ports'
Global frequency set at 100000000000 ticks per second
using riscv64 ISA
warn: Unknown operating system; assuming Linux.
0: system.remote_gdb: listening for remote gdb on port 7001
using LPDDR4X 4266MHz
**** REAL SIMULATION ****
Info: Entering event queue @ 0. Starting simulation...
Info: Increasing stack size by one page.
Info: Increasing stack size by one page.
reset tick: 118190331250
dump tick: 247075678750
Simulated Time : 0.136885 seconds
Exiting @ tick 250355948750 because exiting with last active thread context
rm -rf quantify_result/m5out
mv m5out quantify_result/
root@e1a4699c286c:/home/CA-FP1_quantfly_new/evaluation#
```

2. Default version

```
root@e1a4699c286c:/home/CA-FP1_quantify_default/evaluation# make run_quantify
/home/gen5/build/RISCV/gen5.opt /home/gen5/configs/example/se.py -c quantify.o -o "15" --cpu-type=MInorCPU --cpu-clock=800MHz --mem-type=Ranulator --ramulator-config=/home/gen5/configs/ramulator/LPDDR4-config.cfg --mem-size=256MB --caches --l1i_size=32kB --l1i_assoc=8 --l1d_size=32kB --l1d_assoc=8 --cacheline=32
gen5 Simulator System. http://gen5.org
gen5 is copyrighted software; use the --copyright option for details.

gen5 version 20.1.0.0
gen5 compiled Feb. 7 2022 09:12:04
gen5 started May 20 2025 04:46:48
gen5 executing on e1a4699c286c, pid 2093
command line: /home/gen5/build/RISCV/gen5.opt /home/gen5/configs/example/se.py -c quantify.o -o 15 --cpu-type=MInorCPU --cpu-clock=800MHz --mem-type=Ranulator --ramulator-config=/home/gen5/configs/ramulator/LPDDR4-config.cfg --mem-size=256MB --caches --l1i_size=32kB --l1i_assoc=8 --l1d_size=32kB --l1d_assoc=8 --cacheline=32

warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
warn: membus.slave is deprecated. 'slave' is now called 'cpu_side_ports'
Ranulator system configuration file = /home/gen5/configs/ramulator/LPDDR4-config.cfg
MEMORY TYPE:: Ranulator
Ranulator
warn: membus.master is deprecated. 'master' is now called 'mem_side_ports'
Global frequency set at 100000000000 ticks per second
Using riscv64 ISA
Unknown operating system; assuming Linux.
CDROM: remote_gdb: listening for remote gdb on port 7001
Using LPDDR4X 4266MHz
**** REAL SIMULATION ****
Info: Entering event queue @ 0. Starting simulation...
Info: Increasing stack size by one page.
Info: Increasing stack size by one page.
Reset tick: 2573543391250
Dump tick: 2789416263750
Simulated Time : 0.215873 seconds
Exiting @ tick 2793318175000 because exiting with last active thread context
rm -rf quantify_result/m5out
m5out quantify_result/
root@e1a4699c286c:/home/CA-FP1_quantify_default/evaluation#
```

D. Bonus(Analyze performance-area* tradeoff for Logic Units compared to the Baseline execution on CPU Cores.c)

1. Hardware Parameters

From the paper Near-DRAM Accelerated Matrix Multiplications:

- Area per PU: 0.161 mm²
- Power per PU: 0.0479 W
- Peak Performance per PU: 0.048 TFLOPs

A full-scale configuration with 5120 PUs achieves:

- 245.76 TFLOPs total
- 822.3 mm² total area
- 245.45 W total power

Compared to the Nvidia V100 GPU(125 TFLOPs, 815 mm², 300 W):

- 2× the performance
- ~20% lower power consumption
- Slightly larger area

2. Baseline CPU Comparison

The baseline system provided by the TA is an Intel Xeon E5-2687W v4 @ 3.00GHz with 128 GB DDR4 memory. Although it is a high-end server-grade processor, the lack of specialized hardware support results in relatively higher execution time and power consumption for RNA-related tasks.

Our own local test platform uses an Intel Core i7-13700 (13th Gen) with 32 GB DDR4 memory, configured in a virtual machine with 4 cores and 16 GB of RAM. While this CPU delivers better single-thread performance, it still cannot match the Logic Units in terms of memory locality and latency efficiency.

3. Tradeoff Analysis

In our RNA sequence quantification context, we estimate that Logic Units provide significant acceleration especially for small to medium workloads, which are dominant in bioinformatics applications.

This analysis demonstrates that Near-DRAM architectures are cost-effective and competitive alternatives to traditional CPU or GPU-based systems, especially for data-intensive pipelines.