Streaming Data Systems

Assignment 3

Data Processing Using AIR

Dataflow:

- YSB dataflow acts as the basis for the Dataflow for our task at hand. We modify it a bit and use it to meet our purposes.
- Please note that the different vertices that are used in this dataflow are also an simple extension of the vertices that are present in YSB usecase.

```
RegexMatching::RegexMatching(unsigned long throughput, string pattern) :
        Dataflow() {

    generator = new EventGeneratorRP(1, rank, worldSize, throughput);
    filter = new EventFilterRP(2, rank, worldSize, pattern);
    aggregate = new FullAggregatorRP(3, rank, worldSize);
    collector = new EventCollectorRP(4, rank, worldSize);

    addLink(generator, filter);
    addLink(filter, aggregate);
    addLink(aggregate, collector);

    generator->initialize();
    filter->initialize();
    aggregate->initialize();
    collector->initialize();
}
```

Generator:

- We introduce a new method known as generate_seq which generates a random 50 length string of uppercase letters.

```
string EventGeneratorRP::generate_seq()
{
    const string alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    const int alphabetSize = alphabet.size();
    string sequence = "";
    // sequence +=alphabet;
    // sequence +="ABCEDEFGHIJKLMNOPRSTUVWX";

    std::srand(static_cast<unsigned int>(std::time(0)));

    for (int i = 0; i < 50; ++i)
    {
        sequence += alphabet[rand() % alphabetSize];
    }

    return sequence;
}
```

Filter:

- Once the generator generates the events, they are wrapped into a message and serialized and sent to the next vertex (here that is filter vertex). In the filter vertex, the events are deserialized and are checked if there is any matching pattern in the string.
- We introduce a new function for pattern matching

```
bool EventFilterRP::find_regex(string text, string pattern)
{
    regex regexPattern(pattern);
    smatch match;

    return regex_search(text, match, regexPattern);
}
```

Aggregator:

- The resulting events from the filter are wrapped again into a message and serialized and sent to the next vertex (here that is the aggregator). In the aggregator we desrialize the events and calculate the
    - Total count of events that matched the pattern
    - Window latency

```
    sede.YSBdeserializePCReg(inMessage, &eventPCReg,
            i * sizeof(EventReg));
    sum_latency += eventPCReg.latency;
    count += eventPCReg.count;
```

Collector:

- This is the final stage of our dataflow which outputs the results generated.

```
********************AIR (c) 2020 Uni.lu*******************

AIR INSTANCE AT RANK 1/4 | TP: 100000 | MSG/SEC/RANK: 2 | AGGR_WINDOW: 10000ms
AIR INSTANCE AT RANK 2/4 | TP: 100000 | MSG/SEC/RANK: 2 | AGGR_WINDOW: 10000ms
AIR INSTANCE AT RANK 3/4 | TP: 100000 | MSG/SEC/RANK: 2 | AGGR_WINDOW: 10000ms
AIR INSTANCE AT RANK 4/4 | TP: 100000 | MSG/SEC/RANK: 2 | AGGR_WINDOW: 10000ms
Current time: 390091 Current Window Latest Event Time: 379000

The WID is: 37
Latency of this window: 11091
The number of events in that window id is: 294728
Total latency uptil this window is: 11091
-----------------------------------------------------
Current time: 400087 Current Window Latest Event Time: 389000

The WID is: 38
Latency of this window: 11087
The number of events in that window id is: 360406
Total latency uptil this window is: 22178
-----------------------------------------------------
Current time: 410115 Current Window Latest Event Time: 399000

The WID is: 39
Latency of this window: 11115
The number of events in that window id is: 299748
Total latency uptil this window is: 33293
-----------------------------------------------------
```

Serialization and Deserialization:

- These are the crucial steps of our pipeline as it is essential to maintain the messages flowing in and out of the vertices having the same data type.
- Before passing to the next vertex in the dataflow, each vertex wraps the output set of events into a message and then sent to the next vertex.
- On the other hand, when a vertex receives a message, it first unwraps it back into set of events using the process of deserialization.

```
void Serialization::YSBserializeRG(EventRG* event, Message* message) {
    char* b = message->buffer + message->size;
    memcpy(b, &event->event_time, 8);
    b += 8;
    memcpy(b, &event->ad_id, 51);
    message->size += sizeof(EventRG);
}

void Serialization::YSBdeserializeRG(Message* message, EventRG* event,
        int offset) {
    char* b = message->buffer + offset;
    memcpy(&event->event_time, b, 8);
    b += 8;
    memcpy(&event->ad_id, b, 51);
}
```

Github link:

https://github.com/yashk0311/AIR_Regex

Done by:

Yash Koushik(IMT2020033)