**Міністерство освіти і науки України**

**Національний технічний університет України**

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Теплоенергетичний факультет**

**Кафедра автоматизації проектування  
енергетичних процесів і систем**

***Звіт***

***з лабораторної роботи №2***

***"Розробка конс’юмера даних"***

***з дисципліни***

***"Обробка потокової інформації"***

**Виконав:**

Студент групи ТІ-01мп

Стеценко Д. О.

**Перевірив:**

Д.ф.-м.н.

Матичин І.І.

**м. Київ – 2021 р.**

**Завдання**:

Написати програмну реалізацію алгоритму резервуарної вибірки (Reservoir Sampling). Продемонструвати її роботу на прикладі потоку даних, що генерується продюсером, створеним в рамках Лабораторної роботи №1.

**Передумови**:

Для реалізації конс’юмера даних була використана платформа **Node.js**, бібліотека **Kafka.js**, бібліотека **Rx.js** для створення і генерації потоку даних, та створений у Лабораторній роботі №1 продюсер. Припустимо, що Zookeeper і брокер Kafka вже запущені.

**Хід виконання**:

Спочатку створюємо екземпляр об’єкту kafka з параметрами підключення, потім на основі нього ініціалізуємо consumer, як у лабораторній №2. З коду програми видно, що ми зберігаємо S перших повідомлень, потім замінюємо випадкове збережене повідомлення новим з ймовірністю S / N, де N – це кількість оброблених повідомлень.

Код конс’юмера для отримання даних координат (з частотою .5 секунди) наведений нижче:

const { Kafka } = require('kafkajs');

const { handleErrors } = require('./error-handling');

let processedCount = 0;

const S = 10;

const savedMessages = [];

const kafka = new Kafka({

  clientId: 'coords-app',

  brokers: ['localhost:9092'],

});

const consumer = kafka.consumer({ groupId: 'random-coords' });

const run = async () => {

  console.log('Consumer is running...');

  await consumer.connect();

  await consumer.subscribe({ topic: 'random-coords', fromBeginning: false });

  await consumer.run({

    eachMessage: async ({ *topic*, *partition*, *message* }) => {

      processedCount++;

      if (processedCount < S) {

        savedMessages.push(message.value.toString());

      } else {

        const shouldKeep = Math.random() < S / processedCount;

        if (!shouldKeep) {

          return;

        }

        console.log('> ', message.value.toString());

        const idx = Math.floor(Math.random() \* savedMessages.length);

        savedMessages[idx] = message.value.toString();

      }

    },

  });

};

run().catch((*error*) => {

  console.error(error);

  process.exit(1);

});

process.on('SIGINT', function () {

  consumer.stop();

  console.log('Total processed messages: ', processedCount);

  console.log('Kept messages', savedMessages);

});

handleErrors();

Результат виконання програми:

$ node index.js

Consumer is running...

{"level":"INFO","timestamp":"2ge":"[Consumer] Starting","gro

{"level":"INFO","timestamp":"2ge":"[ConsumerGroup] Consumer emberId":"coords-app-575474cc-p-575474cc-587d-4ab6-93bc-04b6dom-coords":[0]},"groupProtoco

Saving message... { latitude: 6 }

Saving message... { latitude: }

Saving message... { latitude:

}

Saving message... { latitude: }

Saving message... { latitude: 5 }

Saving message... { latitude: Saving message... { latitude: Saving message... { latitude: 41 }

Saving message... { latitude:

}

Saving message... { latitude:

}

Saving message... { latitude: }

Total processed messages: 101

, coordId: 41 },

{ latitude: 49, longitude: 71, id: 'he51ym7q9ym', coordId: 47 },

{ latitude: 76, longitude: 35, id: 'ts6hvsxmyvt', coordId: 71 },

{ latitude: 69, longitude: 104, id: 'x37rshgpk4a', coordId: 84 }

]