Универзитет у Београду

Електротехнички факултет

Мастер 4.0 – Напредне информационе технологије у дигиталној трансформацији

Софија М. Деспотовић, 2019/4002

Јава апликација која за режисере проналази просечну оцену режираних филмова

Дистрибуирани рачунарски системи

01. 09. 2022.

Београд, 2022

[*Задатак 3*](#_Toc112899008)

[*Конкурентно програмирање 4*](#_Toc112899009)

[*Токови података 4*](#_Toc112899010)

[*Удаљени позиви метода (RMI) 4*](#_Toc112899011)

[*Спарк (Spark) 5*](#_Toc112899012)

Задатак

Потребно је написати Јава апликацију која за сваког режисера проналази просечну оцену свих његових режираних филмова (без обзира да ли је та особа режирала читав филм). Излаз апликације је листа која садржи идентификатор режисера и просечну оцену његових снимљених филмова. Подаци о режисерима и сценаристима су доступни у архиви title.crew.tsv.gz. Подаци о оценама филмова су доступне у архиви title.ratings.tsv.gz.

На основу задате формуле, технологије у којима задатак треба да се реши су:

* Коришћење нити, бафера коначног капацитета за размену порука између нити, и баријера за синхронизацију нити.
* Коришћење токова података (java.util.streams)
* Мрежно програмирање користећи удаљене позиве метода, при чему се обрада извршава на неком од удаљених рачунара.
* Користећи технику израчунавања у меморији користећи токове за систем Спарк (Spark)

Конкурентно програмирање

Задатак је решен помоћу мониторске имплементације бафера ограниченог капацитета, као и баријере над којом се синхронизују *Consumer*-и ради слања података даље када заврше.

Помоћу два додатна *Consumer*-a, уводи се конкурентност у обради две различите архиве потребне за добијање решења. Тако, обраду линија добијених од *Producer*-a (који је исти за све типове *Consumer*-a) врше три различита типа *Consumer*-a. Сваки тип *Consumer*-a синхронизује се на сопствену баријеру (која је иста за све типове).

* *ConsumerTitleCrew* – Oбрађује податке из архиве *title.crew.tsv.gz.* За сваког режисера, у бафер уписује податак са називом филма и једним режисером тог филма. Када сви заврше, шаљу своје резултате *Consumer*-у путем бафера који је заједнички само за њих.
* *ConsumerTitleRatings* – Oбрађује податке из архиве *title.ratings.tsv.gz.* У бафер уписује податак са називом филма и оценом тог филма. Када сви заврше, шаљу своје резултате *Consumer*-у путем бафера који је заједнички само за њих.
* *Consumer* – Прво обрађује податке добијене из бафера дељеног са *ConsumerTitleCrew* класом. Потом обрађује податке добијене из бафера дељеног са *ConsumerTitleRatings*. У оквиру саме обраде линија из *ConsumerTitleRatings*, те линије се пореде са претходно обрађеним подацима из *ConsumerTitleCrew*. Разлог за ово је чињеница да података о оценама филмова има знатно мање у односу на податке о режисерима, па ће спаривање режисера са оценом по називу филма тећи брже. Сваком податку (филм, оцена) се придружује припадајући режисер, а затим издваја само податак (режисер, оцена). На основу оваквих прикупљених података, сваки *Consumer* локално нађе тренутни просек оцена за сваког режисера. Када сви заврше, шаљу своје резултате *Combiner*-у.

Обрада се даље наставља у *Combiner*-у, који обавља исти део посла као и *Consumer* како би нашао коначну просечну оцену за сваког режисера на основу парцијалних информација добијених из сваког *Consumer*-а.

Коначно, класа *Printer* излистава сваког режисера, уз његову просечну оцену.

Токови података

Да би се обе архиве обрадиле, коришћена су два паралелна тока чији су резултати еквивалентни резултатима описаним у *ConsumerTitleCrew* и *ConsumerTitleRatings* класама, а чувају се у листама.

Како би се дошло до обједињених резултата, из разлога што података о оценама филмова има знатно мање у односу на податке о режисерима, креира се ток од листе филмова и њихових оцена, како би се сваком филму, на основу имена, додао и режисер. Режисер се додаје креирањем тока у оквиру тренутног тока филмова и оцена.

Када у току имамо све потребне информације (режисер, оцена), оцене групишемо по режисерима, како би их упросечили.

Удаљени позиви метода (*RMI*)

Објекти којима се приступа удаљено су бафер, ради удаљеног слања и примања података са сервера, као и баријера ради синхронизације. За њих се креира интерфес и направе се конкретне имплементације на серверској страни, а затим се упишу у регистар како би могло удаљено да им се приступа.

Ради имплементације баријере и бафера на клијентској страни, користи се *stub* за сваку удаљену референцу објекта. Преко ових имплементација је омогућено да их *Producer*-и, *Consumer*-и, *Combiner*-и и *Printer*-и користе.

Имплементација свих ових класа је идентична као у конкурентном програмирању, уз горепоменуту разлику у позиву бафера и баријера и могућност спецификације на ком *host*-у и *port*-у се налазе објекти којима приступамо.

Спарк (*Spark*)

Да би се обе архиве обрадиле, коришћена су два *Spark* тока чији су резултати еквивалентни резултатима описаним у *ConsumerTitleCrew* и *ConsumerTitleRatings* класама, а чувају се у парним подацима типа *Resilient Distributed Datasets* (*RDD*), коришћеним јер пружају подршку за податке типа кључ-вредност.

Кључ је у овом кораку назив филма, и служи како би могли да креирамо нови сет података користећи методу *join()* – помоћу које се на основу кључа сједињују вредности из *titleRatings* и *titleDirectors* сетова података. Самим овим кораком, редукује се број података о филмовима и режисерима, јер уколико постоје филмови у једном сету података, али не у другом, требало би да се одбаце сувишни.

Напослетку, како би знали број филмова по режисеру, уводи се и бројач оцена за режисера, и просек се добија дељењем сумарне вредности оцена филмова сумарном вредношћу броја оцена.