**Универзитет у Београду**

**Електротехнички факултет/Факултет организационих наука**

Logo

Description automatically generated

**Јава апликација –**

**Режисери и укупни минути њихових режираних филмова**

**Предмет:** Дистрибуирани рачунарски системи

**Студијски програм:** Напредне информационе технологије

у дигиталној трансформацији

|  |  |
| --- | --- |
| **Професор:** | **Студент:** |
| **Захарије Радивојевић** | **Бранкица Јовановић 4011/2019** |

*Прва верзија:29.01.2021.*

*САДРЖАЈ*

1. [***Задатак***](#zadatak) ***3***
2. [***ТЕХНОЛОГИЈЕ***](#tehnologije) ***3***

*[2.1.](#barijera)**[Коришћење нити, бафера коначног капацитета за размену порука између нити и баријера](#barijera)*

*[за синхронизацију нити](#barijera) 4*

[*2.2.* *Коришћење токова података (java.util.streams)*](#strimovi) *5*

[*2.3. Мрежно програмирање користећи прикључнице (socket)*](#soketi) *6*

[*2.4. Коришћење технике мапирања и редукције (MapReduce) за систем Hadup (Hadoop)*](#hadup) *7*

1. **ЗАДАТАК**

Потребно је написати Јава апликацију која за сваког режисера проналази колико су укупно минута трајали сви његови режирани филмови (без обзира да ли је та особа режирала читав филм). Излаз апликације је листа која садржи идентификатор режисера и број минута снимљених филмова. Подаци о филмовима/серијама су доступни у архиви title.basics.tsv.gz. Подаци о режисерима и сценаристима су доступни у архиви title.crew.tsv.gz.

1. **ТЕХНОЛОГИЈЕ**
2. Коришћење нити, бафера коначног капацитета за размену порука између нити, и баријера за синхронизацију нити.
3. Коришћење токова података (*java.util.streams*)
4. Мрежно програмирање користећи прикључнице (*енг. socket*), апликацију на којој се налази сервер и већи број радних станица које раде обраду.
5. Техника мапирања и редукције (*енг. MapReduce*) за систем Хадуп (*енг.* *Hadoop*)
   1. **КОРИШЋЕЊЕ НИТИ, БАФЕРА КОНАЧНОГ КАПАЦИТЕТА ЗА РАЗМЕНУ ПОРУКА ИЗМЕЂУ НИТИ И БАРИЈЕРА ЗА СИНХРОНИЗАЦИЈУ НИТИ**

Имплементација коришћење нити, бафера коначног капацитета за размену порука између нити и баријера за синхронизацију нити налази се у класама *Main,* *Producer, Buffer, Consumer, Combiner, Barrier i Printer*.

*Hash map-a trajanjeFilmova,* са кључем идентификатор филма и вредношћу минути трајања филма, се кешира у меморији и учитава по потреби.

Ha *Slici 1.* приказана је шема функционалне имплементације. *Producer* учитава фајл и избацује листу задатака за *Consumer*-a. Сваки задатак садржи једну линију фајла тј. један филм. С обзиром на количину филмова и података, *Buffer* контролише тј. пропушта онолико филмова колико задати број *Consumer-*аможе да обради. *Consumer* очитава филм по филм, проверава идентификатор филма у *Трајању филмова* и прави објекат који у себи садржи низ режисера и минуте. Сви режисери једног филма се један по један (заједно са минутима) шаљу *Combiner*-у. Када је улазни *Buffer* празан, *Consumer-*и чекају на *Barrier-*y*,* а затим се шаље сигнал за крај ка *Combiner*-у*. Combiner* прави јединствени низ - режисер и збирни минути, тако што сабира минуте јединственог појединачног режисера. На крају се све прослеђује *Printer*-у на штампање.

A white board with writing on it

Description automatically generated with low confidence

*Слика 1. Шема функционалне имплементације*

* 1. **КОРИШЋЕЊЕ ТОКОВА ПОДАТАКА (java.util.streams)**

Имплементација коришћење токова података налази се у класи *MainStreamResenje*.

*Hash map-a trajanjeFilmova,* са кључем идентификатор филма и вредношћу минути трајања филма, се кешира у меморији и учитава по потреби.

Ha *Slici 2.* приказана је шема функционалне имплементације.Улазни фајл се ишчитава и претвара у ток података који садржи податке о филмовима. Линије су претворене у ток података објекат типа *Režiseri* који садржи члан идентификатора филма и режисере у текстуалној репрезентацији. Затим се коришћењем функције *map* прави други ток података објекат типа *RežiseriNiz* који садржи чланове низ идентификатора филма и низ одвојених режисера. Трећи ток података, представља низ објеката са вредностима један идентификатор филма и режисер, а затим се врши пресликавање *1 → n* коришћењем функције *flatmap,* где од низа објеката добијамо ток података од објекта са идентификатором филма и једним режисером који није неопходно јединствен. У следећем току података из објекта се проверава идентификатор филма у *Трајању филмова* и прави ток података од објекта који у себи садржи низ идентификатор, режисери и минути у коме такође режисер може да се понавља. На крају се коришћењем функције *groupby* прави јединствени низ - режисер и збирни минути, тако што се сабирају минути јединственог појединачног режисера.

Наведеном начину имплементације било је потребно најкраће време за добијање резултата.

Diagram

Description automatically generated

*Слика 2. Шема функционалне имплементације*

* 1. **МРЕЖНО ПРОГРАМИРАЊЕ КОРИСТЕЋИ ПРИКЉУЧНИЦЕ (engl. SOCKET)**

Имплементација мрежног програмирања користећи прикључнице (*енг. socket*), апликацију на којој се налази сервер и већи број радних станица које раде обраду, налази се у класама *MainProducerCombinerPrinter,* *MainConsumer, ServerBuffer, Producer, Buffer, BufferNet, Consumer, Combiner, Barrier, BarrierNet i Printer*.

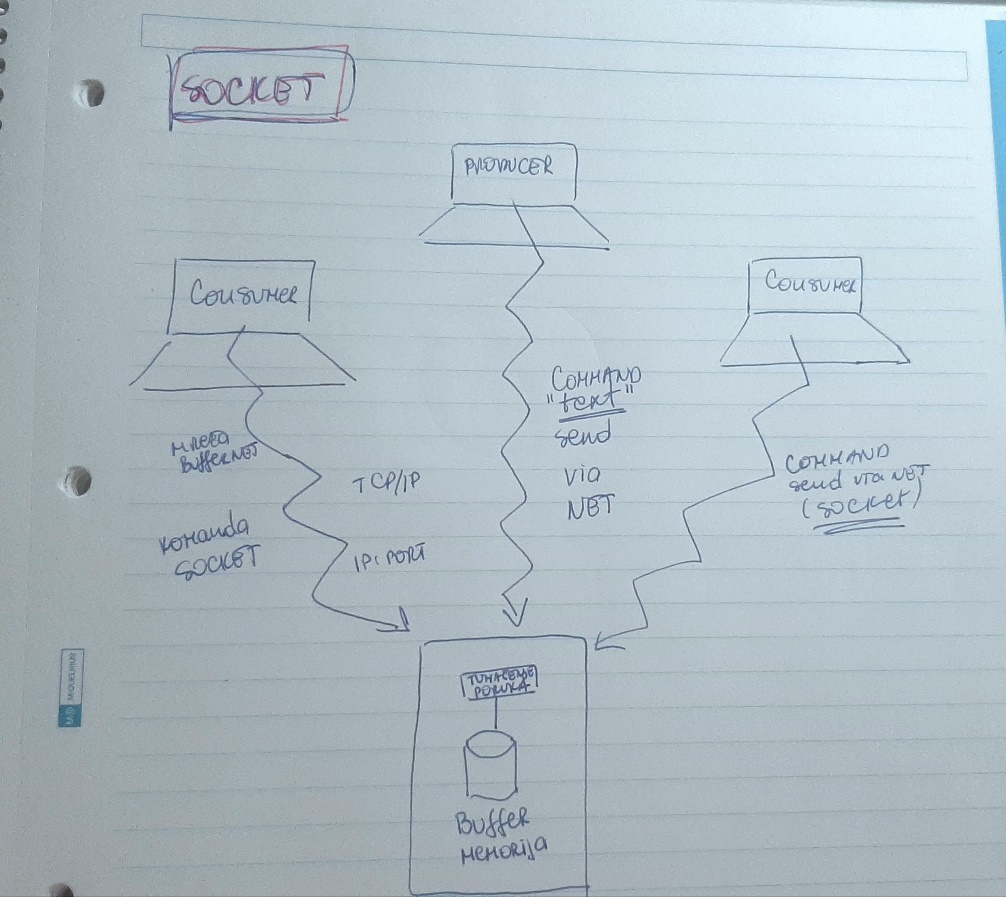
Ha *Slici 3.* приказана је шема функционалне имплементације.Циљ је да се симулира коришћење више рачунара на мрежи за решавање задатог проблема. Пошто нам је на располагању један рачунар, симулација значи да користимо и покренемо више процеса на једном компјутеру, где сваки од процеса предстаља улогу коју би имао посебан рачунар на мрежи.

Улоге су следеће: *ServerBuffer* за размену порука, *Producer* за давање задатака, *Consumer* за обраду задатака, *Combiner* за склапање финалног резултата и *Printer* за штампање излаза. Повећањем броја *Consumer-а* у пракси (на различитим рачунарима) би се пропорционално постигло убрзавање извршавања.

Највећа имплеметациона разлика у односу на *задатак број 2.1*, је у коришћењу прикључница за размену порука између клијената и сервера на коме се користи идентична имплементација баријере и бафера као у наведеном задатку.

Укратко, једна линија филма се од *Producer-а* шаље на *ServerBuffer* (*put* текстуална команда са назнаком имена бафера), а *Consumer*-и је узимају један по један (*get* текстуална команда са назнаком имена бафера). Аналогно се одиграва разговор измеју *Consumer*-а и *Combiner*-а који интерно обављају исти посао као у *задатак број 2.1*. *Combiner* на крају шаље серверу (*put*) припремљене резултате збирних минута по јединственом режисеру и *Printer* их крају штампа након преузимања, коришћењем *get* команде.

На локалном компјутеру, повећање броја консумера само донекле убрзава (до 2), а преко 2 због ефекта презасићења процесора од 2 језгра се извршавање успорава (што је и очекиванo).



*Слика 3. Шема функционалне имплементације*

* 1. **ИМПЛЕМЕНТАЦИЈА 4 – ХАДУП**

Имплементација коришћењем техника мапирања и редукције (енг. *MapReduce*) за систем Хадуп (енг. *Hadoop*) налази се у класи *ReziseriMinuti*.

*Hash map-a trajanjeFilmova,* са кључем идентификатор филма и вредношћу минути трајања филма, се кешира у меморији и учитава по потреби.

Ha *Slici 4.* приказана је шема функционалне имплементације.Имплементиране су *Map* и *Reduce* компоненте (не улазећи у детаљну структуру Хадуп-а), а остатак система посматрамо као већ функционалан уз минималне измене.

Улаз је фајл у коме је један ред – један филм. Расподела улазних података је већ подржана и *Map* добија на улазу један ред као вредност. Аналогно као у познатом *WordCount* примеру, улазни ред се претвара у речник где је кључ идентификатор режисера*,* а вредност трајање филма односно минути. Трајање филма добијамо, као и пре, из табеле учитане у меморију за потребу брзог приступа *trajanjeFilmova*.

Припремом речника (кључ: режисер, вредност: трајање), посао *Mapper* -а се завршава и подаци се прослеђују *Hadoop*-у на *schuffle* процес и онда се притупа *Reduce* поступку.

*Reduce* већ добија груписане податке по режисеру и има једноставан задатак да сумира трајање тј. збирне минуте. На крају се штампа финални резултат у посебан излазни фајл (већ имплементиран као део *Hadoop*-а) .

A picture containing text, whiteboard

Description automatically generated

*Слика 3. Шема функционалне имплементације*