



Електротехнички факултет
Универзитет у Београду

Конкурентно и дистрибуирано програмирање
**Дистрибуирана обрада
математичких израза**

Прва верзија: 01. јун 2009. године
Текућа верзија: 24. јун 2009. године

Студент:
Марко Јовановић 05/243

Београд

САДРЖАЈ:

ТЕКСТ ЗАДАТКА	3
ОПИС ПРЕДЛОЖЕНОГ РЕШЕЊА	3
ДИЈАГРАМ ИНТЕРАКЦИЈЕ.....	5
УПУТСТВО ЗА КОРИШЋЕЊЕ СИСТЕМА	6
УПУТСТВО ЗА КОРИШЋЕЊЕ СИСТЕМА У ЦИЉУ ДАЉЕГ УСАВРШАВАЊА	6
ПРИМЕР РАДА СИСТЕМА У РЕГУЛАРНИМ И НЕРЕГУЛАРНИМ СИТУАЦИЈАМА.....	7
ТЕСТОВИ ПЕРФОРМАНСИ.....	7

ТЕКСТ ЗАДАТКА

Пројектовати дистрибуирани рачунарски систем који треба да омогући дистрибуирану обраду и међусобну удаљену синхронизацију временски захтевних послова математичке обраде.

Програм треба да ради у систему који се састоји од више рачунара повезаних у LAN (Local Area Network) или WAN (Wide Area Network).

У систему постоји три типа програма:

1. Централни сервер који служи за праћење рада извршавања дистрибуиране обраде, чување информација о доступним чворовима у мрежи и могућност поновног стартовања појединих послова.

2. Радна станица која од централног сервера добија послове које треба одрадити.

3. Кориснички програм, који задаје посао који треба урадити и његове параметре.

Процес започиње тако што кориснички програм задаје посао који је потребно извршити. Овај посао представља математички израз који је потребно израчунати. Након тога кориснички програм контактира централни сервер коме прослеђује тај посао и параметре потребне за његову обраду. Када централни сервер прими посао и његове параметре он га прослеђује једној радној станици, чека резултат обраде и враћа комплетан резултат клијенту. Када радна станица прими посао, креира нит намењену извршавању тог посла, и у тој нити започиње са његовим извршавањем на основу примљених параметара. Посао се извршава на радној станици тако што покренута нит радне станице позива одговарајућу методу инстанце класе

за паралелно процесирање математичких изрази (имплементира интерфејс *Equations*). Да би се обављало паралелно процесирање потребно је прво иницијализовати ову инстанцу адресама и портovima преосталих класа за паралелно израчунавање који се налазе на осталим радним станицама (интерфејс *Connector*), програму за паралелно израчунавање је потребно доделити један слободни серверски порт (*setPort*) и покренути их свакој радној станици (метод *connect()*). Инстанцирање класа које раде математичку обраду и повезивање на радној станици обезбеђује се користећи статичке методе класе *Creator*. Када се заврши израчунавање математичких изрази, радна станица прикупља излаз рачунања, пакује га у

одговарајућу датотеку коју прослеђује серверу, као и резултате извршавања. Да би се обезбедило прикупљање информација о активним радним станицама централни сервер на сваких x секунди проверава да ли је нека радна станица исправна. Уколико сервер утврди да нека радна станица која је до тог тренутка обављала математичка израчунавања није више исправна, посао који је обављала та радна станица прослеђује некој слободној радној станици. Након слања захтева за обраду кориснички програм може да раскине везу са

централним сервером. Веза може бити раскинута гашењем програма или затварањем комуникационог канала. Када се следећи пут повеже кориснички програм може да тражи резултате претходно задате обраде. Треба обезбедити да централни сервер може да у паралели да прима већи број послова које је потребно обрадити. Кориснички програм може од централног сервера да тражи информације о статусу посла, а може да тражи и резултате. Одмах по стартовању радне станице шаљу централном серверу информацију о томе да су стартоване и број послова које могу у паралели да обрађују. Број послова које радне станице могу да приме је аргумент који им се поставља приликом покретања.

Параметри посла које клијент задаје су: математичка операција коју је потребно извршити, датотека у којој се налази корисникова матрица коју је потребно обрадити, датотека са низом вредности које је потребно још применити у операцији (параметар је опциони), име датотеке у коју је потребно сместити резултате операције коју је са радне станице треба пребацити на клијентски рачунар да које представљају резултате. Ови параметри могу да се задају или путем корисничког интерфејса или путем текстуалне датотеке.

Централни сервер у лог уписује време када је пристигао сваки посао, број под којим је посао сачуван, име рачунара коме је посао прослеђен, време када је посао завршен и његов тренутни статус. Статус посла може да буде: Ready – приспео на сервер али није никоме прослеђен, Scheduled – тренутно се прослеђује радној станици, Running – извршавање је у току, Done – посао се успешно извршио, Failed – посао није могао да се изврши (емитовао је неки изузетак), Aborted – корисник је одустао од извршавања посла.

Проблем решити користећи мрежну комуникацију у програмском језику Јава. Решење треба да буде независно од посла који се обавља. За сваки од ова три типа рачунара треба да постоји одговарајући графички кориснички интерфејс (GUI треба да буде развијен користећи Јава **SWING** компоненте). Радна станица треба да има могућност покретања и без корисничког интерфејса.

ОПИС ПРЕДЛОЖЕНОГ РЕШЕЊА

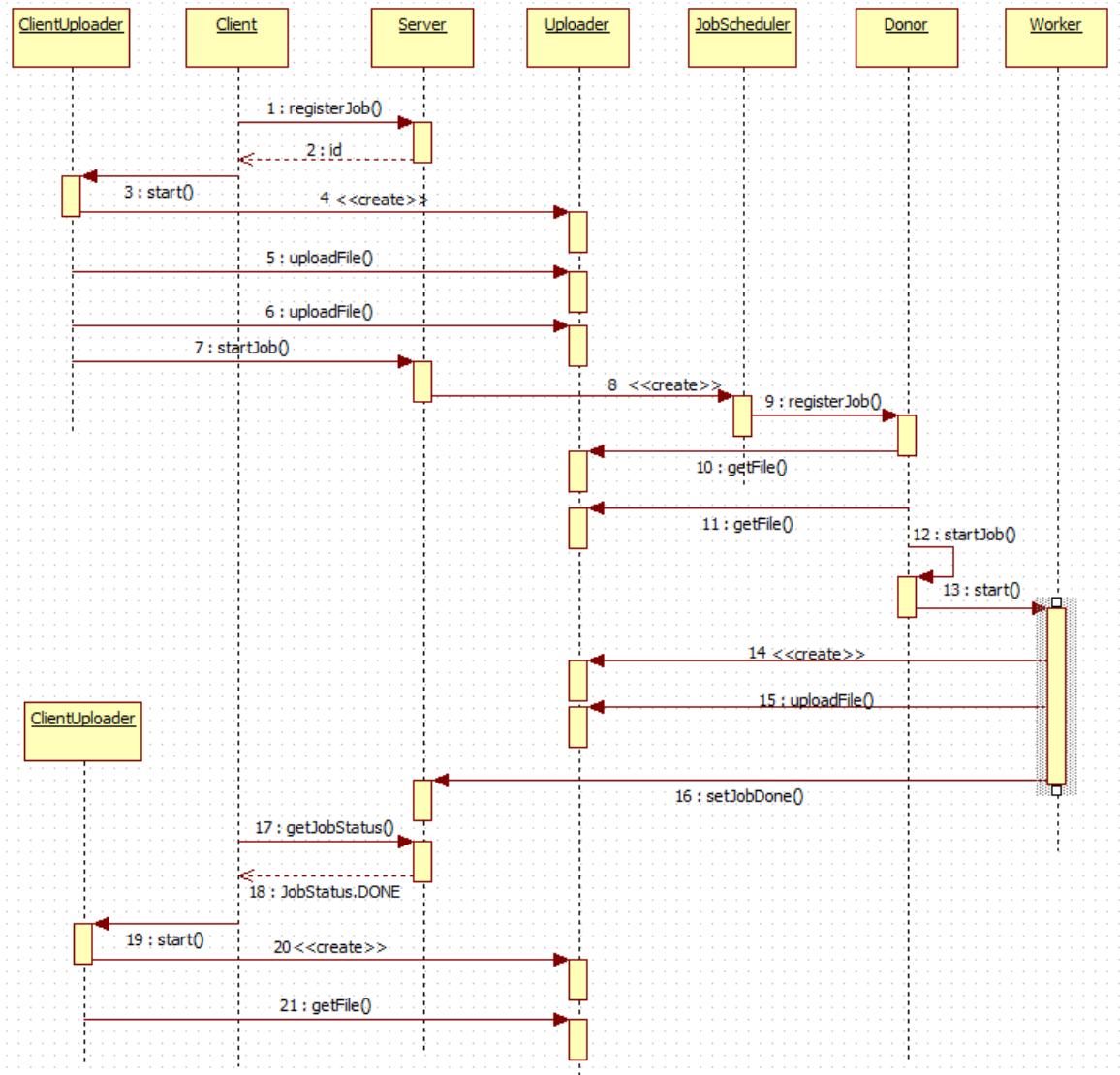
У систему постоје три врсте ентитета: клијент, сервер и радна станица (понекад названа и донор, што је равноправни назив у даљем тексту). Сваки је представљен по једном класом која имплементира основне функције тих ентитета. Комуникација између њих је остварена помоћу позива удаљених процедура (кориштен је Java RMI).

Сервер (у класи Server) чува листу свих приспелих послова на серверу (информације о пословима су имплементирани у виду класе JobInfo). Сервер такође чува информације о пријављеним радним станицама,

колики капацитет су пријавиле да могу истовремено послова да извршавају и о броју послова које тренутно извршавају. Позивом одговарајућих метода клијент креира нови посао, шаље потребне фајлове на сервер и на крају серверу јавља (тек пошто је послао све информације) да посао покрене. Уколико има радних станица које одмах могу да прихвате посао, сервер региструје на једној таквој посао (све позивом одређених метода радне станице преко референце коју је добио приликом регистрације те радне станице на серверу). Започињање посла на радној станици резултује тиме да дозор скида све неопходне датотеке са сервера, смешта их у свој локални директоријум и креира нову нит за обраду тог посла која позива одговарајуће методе из инстанце класе за паралелно процесирање математичких израза.

Уколико приликом пристизања посла на сервер нема слободних радних станица на располагању, посао чека све док нека од радних станица не постане слободна и то јави серверу, или док не пристигне нека нова слободна радна станица. Посебна нит на серверу на сваких 5 секунди пролази кроз списак радних станица и прозива их. Ако је нека станица недоступна, она се брише из система а сви послови за које је сервер знао да је извршавала до тог тренутка, или је посредно (потенцијално) извршавала тако што је била активна у тренутку када је некој другој радној станици додељиван посао, тада се сви послови са такве станице додељују некој другој радној станици уколико има слободних радних станица или чекају док се не ослободи нека од њих одн. пристигне нова радна станица, на начин који је описан.

ДИЈАГРАМ ИНТЕРАКЦИЈЕ



УПУТСТВО ЗА КОРИШЋЕЊЕ СИСТЕМА

У систему је прво потребно подићи сервер. Сервер се подиже покретањем server.bat фајла на Windows платформи или задавањем еквивалентне команде Java виртуелној машини:

```
java -Djava.security.policy=java.policy -  
jar server.jar
```

У пољу Server files потребно је задати путању до директоријума који ће се користити за смештање датотека који представљају улазе и излазе сваког посла који се обрађује на серверу. Сервер ће током свог рада у овом директоријуму за сваки посао креирати директоријум под именом jobID где је ID јединствени идентификатор посла додељен од сервера приликом креирања посла.

У поље Port је потребно уписати један слободан и firewall-ом незаштићен порт на коме ће бити креиран Java RMI Registry.

Притиском на дугме Start сервер започиње са радом. До његовог гашења, сервер ће од корисника прихватити захтеве за извршавање послова и упите о стању послатих послова.

Server поред осталог води и лог који се налази на путањи rootDir\log.txt где rootDir представља већ поменути директоријум на серверу у коме ће се чувати фајлови. У логу се могу наћи информације о пословима односно време када су они пристигли на сервер или прешли у неко друго стање чиме се додатно може пратити исправност система. Лог је могуће читати и након заустављања система као и за време рада система.

Независно од тога да ли је нека радна станица већ пријављена у систем, клијент може у сваком тренутку да шаље послове. Клијент се покреће помоћу фајла client.bat на Windows платформама, или задавањем еквивалентне команде Java виртуелној машини:

```
java -Djava.security.policy=java.policy -  
jar client.jar
```

У поље Server клијента потребно је унети IP адресу или host name рачунара на коме је покренут сервер. У поље Port је потребно уписати порт сервера на коме је креиран Java RMI Registry.

Command file представља датотеку у којој се налазе називи фајлова који представљају улаз посла који треба урадити и који ће бити послати на сервер као и команду коју треба извршити, а према задатој спецификацији.

Такође је могуће мануелно задати путање до неопходних датотека у одговарајућим пољима, као и назив датотеке у коју треба сместити резултат. Поред тога, неопходно је и навести и коју операцију треба извршити.

Кликом на дугме Start Job се покреће процес слања датотека на сервер и покретање посла. Уколико је комуникација са сервером исправно успостављена, статусна лабела ће исписати универзални идентификациони број посла који му је сервер доделио. Овај идентификатор се касније може користити за проверу статуса посла и за прикупљање датотеке са резултатима од сервера.

Дугметом get status се проверава статус посла чији је идентификатор задат. Лабела исписује опис тог статуса.

Дугметом get result се иницира повлачење датотеке са резултатом чији је идентификатор задат.

Дугметом abort job се сервер обавештава да треба да прекинути даљи рад на послу чији је идентификатор задат.

Радном станицом се може управљати помоћу фајла workstation.bat на Windows платформама, или задавањем еквивалентне команде Java виртуелној машини:

```
java -Djava.security.policy=java.policy -  
jar donor.jar
```

У поље Server клијента потребно је унети IP адресу или host name рачунара на коме је покренут сервер. У поље ServerPort је потребно уписати порт сервера на коме је креиран Java RMI Registry.

У поље OwnPort је потребно уписати сопствени серверски порт преко кога ће тећи комуникација између библиотека за паралелно рачунање на самим донорима.

У поље max jobs је потребно уписати максималан број послова које та радна станица може истовремено да прихвати на обраду.

УПУТСТВО ЗА КОРИШЋЕЊЕ СИСТЕМА У ЦИЉУ ДАЉЕГ УСАВРШАВАЊА

Идентификациони бројеви послова у систему су јединствени и додељује их сервер редом по редоследу пристизања од клијената, почев од нуле. Идентификациони број посла се враћа клијенту, како би касније клијент могао да прочита статус посла са сервера и да довуче датотеку са резултатом, у колико је посао успешно окончан. Одмах након регистрације посла на серверу, клијент у посебној нити шаље неопходне датотеке серверу. Посебна нит се користи како би се избегло блокирање графичког корисничког интерфејса. Аналогно, приликом довлачења датотеке са резултатом са сервера, клијент ће креирати посебну нит за ту намену.

Како би се спречила грешка у случају неисправног посла (нпр. неисправна команда у командној

датотеци)

Креирање посла на серверу резултује креирањем директоријума за тај посао у коме ће се смештати неопходне датотеке, као и покретање нити која ће покушати да распореди посао на неку од расположивих радних станица.

У колико нит за распоређивање пронађе адекватну радну станицу, покреће се метода за регистрацију посла на таквој радној станици и њено стартовање. Тада уједно и нит за распоређивање послова умре и неопходно је поново креирати и стартовати нову нит за распоређивање послова при следећем приспећу новог посла.

Приликом тражења датотеке са резултатима од сервера, клијент ће најпре испитати статус посла на серверу, и тек ако је посао успешно завршен, покренути нит која ће довући датотеку са резултатом, како не би дошло до блокирања графичког корисничког интерфејса.

Сервер из листе радних станица избацује оне које су позвале методу за deregистрацију. Ову методу станице позивају приликом заустављања. Уколико је радна станица нерегуларно угашено, ова метода неће бити позвана од стране радне станице. Од нерегуларног рада у овом случају као и у случају када радна станица није доступна из других разлога (прекид везе између радне станице и остатка мреже, сервера и мреже итд...) сервер се брани тако што по свом старту покреће нит која на сваких 5 секунди пролази кроз листу регистрованих радних станица и проверава да ли су доступни. Уколико током тог позива за проверу дође до изузетка (станица је недоступна), она се брише из листе расположивих радних станица, остале доступне радне станице се о томе обавештавају а сервер, у колико је она извршавала неки посао (било директно, било посредно-потенцијално), њен посао покушава нова нит за распоређивање послова да додели некој од слободних радних станица.

ПРИМЕР РАДА СИСТЕМА У РЕГУЛАРНИМ И НЕРЕГУЛАРНИМ СИТУАЦИЈАМА

Регуларна ситуација: подигнути су сервер и радна станица и комуникација између њих је успостављена. Клијент који има комуникацију са сервером шаље серверу посао. Сервер прима податке и након успешног пријема свих факова, креира посао на радној станици и прослеђује јој цео посао. Након креирања сервер клијенту враћа јединствени идентификатор посла. Када радна станица заврши цео посао, јавља серверу да је он готов и шаље серверу датотеке са називима задатим дефиницијом посла. Клијент за све ово време може да проверава статус посла. Након што тај статус постане Done, клијент може да скида датотеку са резултатом са сервера.

Нерегуларна ситуација: Све исто као у предходној тачки са разликом да датотека коју клијент жели да

пошаље на сервер не постоји на задатој локацији. У том случају се прекида даље слање датотеке на сервер а посао добија статус Aborted. Такав посао се неће извршавати на радним станицама.

ТЕСТОВИ ПЕРФОРМАНСИ

Ради мерења перформанси ситема, извршен је тест рачунања инверзне матрице димензије 150x150. Такав тест се извршавао увек око 10 секунди, без обзира на број радних станица. Примећено је да не постоји комуникација између радних станица те да нешто са библиотеком за паралелно рачунање математичких операција нешто није у реду.