

Универзитет у Београду
Електротехнички факултет
Катедра за рачунарску технику и информатику



Дистрибуирана обрада математичких израза

домаћи задатак из Конкурентног и дистрибуираног програмирања

студент:
Саша Поповић
2005/0278

прва верзија: 2.6.2009.
текућа верзија: 25.6.2009

Београд

Дистрибуирана обрада математичких израза

Садржај:

<i>Текст задатка.....</i>	<i>3</i>
<i>Опис предложеног решења.....</i>	<i>3</i>
<i>Сервер.....</i>	<i>3</i>
<i>Клијент.....</i>	<i>4</i>
<i>Радна станица.....</i>	<i>4</i>
<i>Имплементациони опис предложеног решења.....</i>	<i>5</i>
<i>Опис система UML дијаграмима интеракције.....</i>	<i>6</i>
<i>Упутство за коришћење.....</i>	<i>8</i>
<i>Упутство за коришћење у циљу даљег усавања.....</i>	<i>8</i>
<i>Пример рада програма у регуларним и ванредним ситуацијама.....</i>	<i>8</i>
<i>Тестови перформанси.....</i>	<i>9</i>

ТЕКСТ ЗАДАТКА

ДОМАЋИ ЗАДАТАК ЈУН 2009

ДИСТРИБУИРАНА ОБРАДА МАТЕМАТИЧКИХ ИЗРАЗА

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 израза (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 израза, 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 речити 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

Систем за дистрибуирану обраду математичких израза се састоји од три целине: *Клијент*, *Сервер* и *Радна станица*.

СЕРВЕР

Сервер је централни део система. Клијенти серверу шаљу послове које треба извршити. Сервер може у паралели примати захтеве од више клијената. По

примању захтева за извршење посла сервер даје клијенту идентификацију посла на основу које ће клијент касније тражити информације о статусу као и саме резултате. Сервер затим од клијента преузима улазне податке за посао. Преузети послови се распоређују на радне станице на којима има расположивих ресурса. Ако таквих нема, послови се стављају у ред за чекање. Сервер чува и листу радних станица и периодично проверава да ли су оне и даље активне. Уколико је комуникација са неком радном станицом прекинута, сервер обавештава све активне станице о томе, а затим рестартује све незавршене послове које је та радна станица могла да извршава. Посао се сматра завршеним тек онда када се комплетан резултат посла пребаци са радне станице на сервер.

КЛИЈЕНТ

Клијент серверу шаље послове које треба извршити. Пре слања захтева клијент врши валидацију самог посла (проверава да ли су сви параметри захтеваног посла коректни) па некоректно задате послове није могуће послати на извршавање. На основу идентификације коју добија од сервера, клијент може да тражи информације о тренутном статусу посла који је задао као и резултате посла уколико је он успешно извршен. Клијент чува идентификације тражених пословима на чврстом диску, тако да он између слања захтева и прикупљања резултата не мора бити у вези са сервером нити мора бити активан.

РАДНА СТАНИЦА

Радна станица се по свом стартовању повезује на сервер. Она га обавештава о броју послова које може да извршава истовремено и сервер радну станицу неће запослити са више послова него што она може да извршава у паралели. Уколико се то ипак догоди, радна станица прекобројни посао неће прихватити. Нако прихватања посла радна станица преузима са сервера све потребне улазне податке за посао који треба да обави, извршава посао и затим се се јавља серверу. Он преузима резултате од радне станице и тек тиме је један посао у потпуности завршен. Уколико извршавање посла не успе из било ког разлога радна станица обавештава сервер да извршавање посла није успело. Када клијент буде тражио статус овог посла добиће информацију да посао није могао бити извршен, па клијент може поново тражити извршавање тог посла уз евентуалну претходну корекцију параметара посла.

ИМПЛЕМЕНТАЦИОНИ ОПИС ПРЕДЛОЖЕНОГ РЕШЕЊА

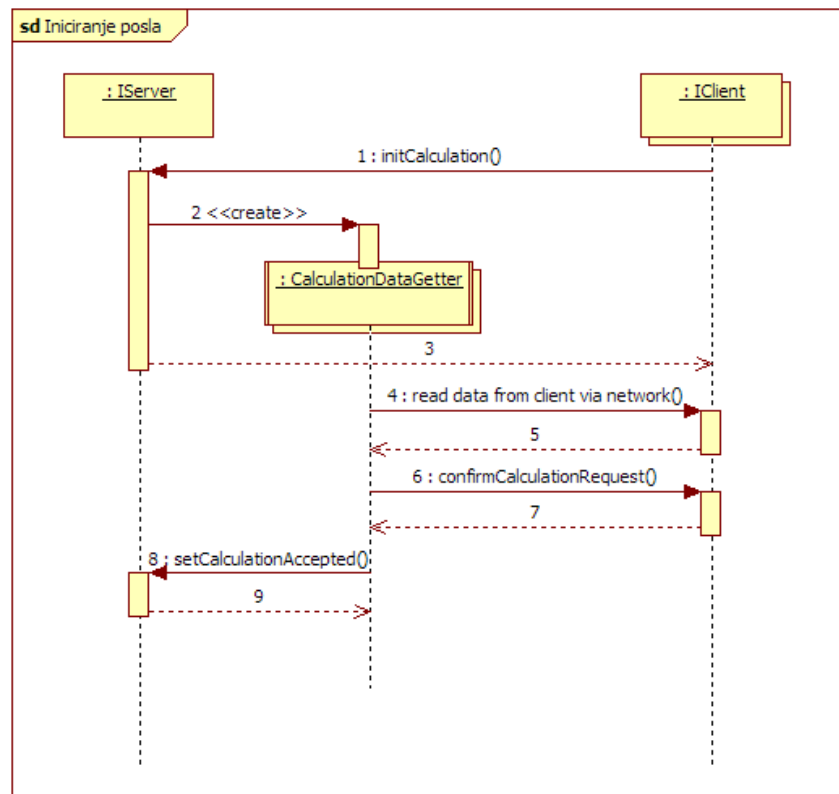
Језгро решења задатка се налази у класама *Server*, *Client* и *Workstation*, које се међусобно “виде” преко интерфејса *IServer*, *IClient* и *IWorkstation*. Оне имплементирају и енкапсулирају сву потребну комуникацију и синхронизацију. Комуникација између удаљених рачунара је реализована помоћу позива удаљених процедура за шта је коришћен пакет *Java Remote Method Invocation*. За пренос датотека између удаљених рачунара коришћен је пакет *RMIO*.

Класа *Server* по свом стартовању покреће две нити. Једна од њих, реализована класом *Disconnected-WorkstationFinder*, периодично проверава да ли су све пријављене радне станице станице и даље активне и предузима потербне мере уколико утврди да је комуникација са неком од радних станица у прекиду. То подразумева да обавештава све активне радне станице о прекиду везе и да рестартује све послове које је радна станица са којом је веза прекинута могла да извршава. Друга нит, реализована класом *CalculationScheduler*, распоређује послове по радним станицама и она се блокира када нема нераспоредених послова или када нема слободних радних станица. Свако пристизање новог посла, пријављивање нове радне станице или завршетак неког посла активирају ову нит. *Server* је задужен да пословима даје јединствене целобројне идентификације, а потребне податке о пословима (енкапсулиране у класу *Calculation*) чува у хеш табели (кључ представља идентификација посла). Временски захтевни послови и послови за које је захтевано да се могу извршавати у паралели, као што су довлачење улазних фајлова за послове са клијентских рачунара и дохватање резултата са радних станица се стартују у засебним нитима које су реализоване класама *CalculationDataGetter* и *CalculationResultGetter*. У тренутку када радна станица започиње извршавање неког посла она о томе обавештава сервер, како би он листу активних радних станица у том тренутку забележио у податке о послу који се стартује, да би касније у случају пада везе са било којом станицом из те листе посао могао да рестартује и тиме обезбеди да се посао изврши.

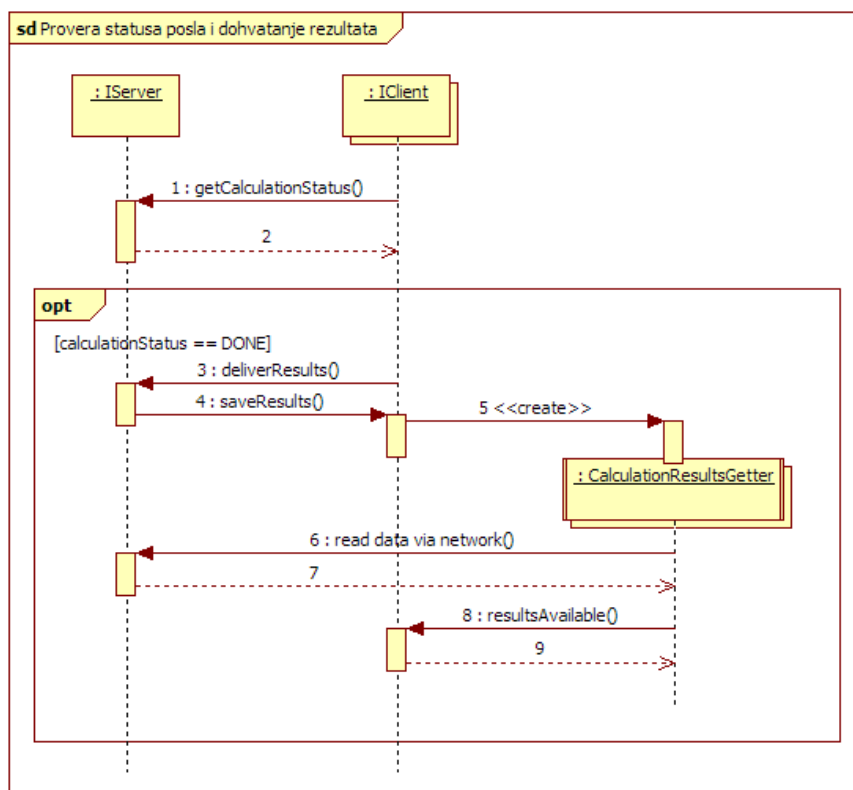
Класа *JFServer* представља визуелни кориснички интерфејс сервера преко које сервер може да се конфигурише и преко које може да се прати стање сервера. Комплетна логика функционисања се ослања на већ описану класу *Server*, која у случају да има референцу на класу *JFServer* исту обавештава о догађајима чиме се подаци о стању сервера редовно ажурирају на корисничком интерфејсу.

Инстанце класа *Client* и *Workstation* је потребно иницијализовати информацијама о серверу (IP адреса, порт, име под којим ће се препознати објект сервера) како би се могла успоставити комуникација преко мреже, а затим оне прослеђују параметре преко којих ће сервер контактирати њих.

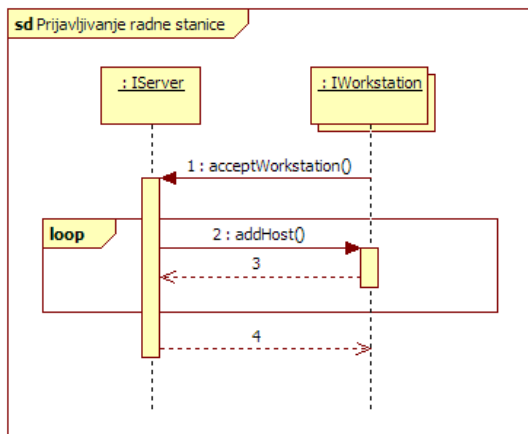
Као што је случај и са серверском страном и за клијента и за радну станицу постоји графички кориснички интерфејс (*JFClient* и *JFWorkstation*) који који сву логику посла који се обавља реализује ослањајући се на класе *Client* и *Workstation*. У наставку следи приказ дијаграма интеракције између клијента и сервера као и између радне станице и сервера.



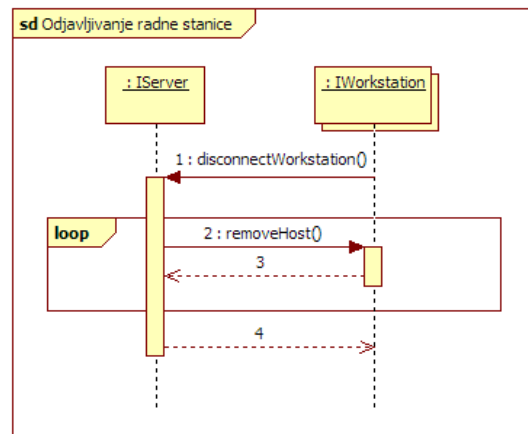
Дијаграм интеракције: Иницирање посла (клијент - сервер)



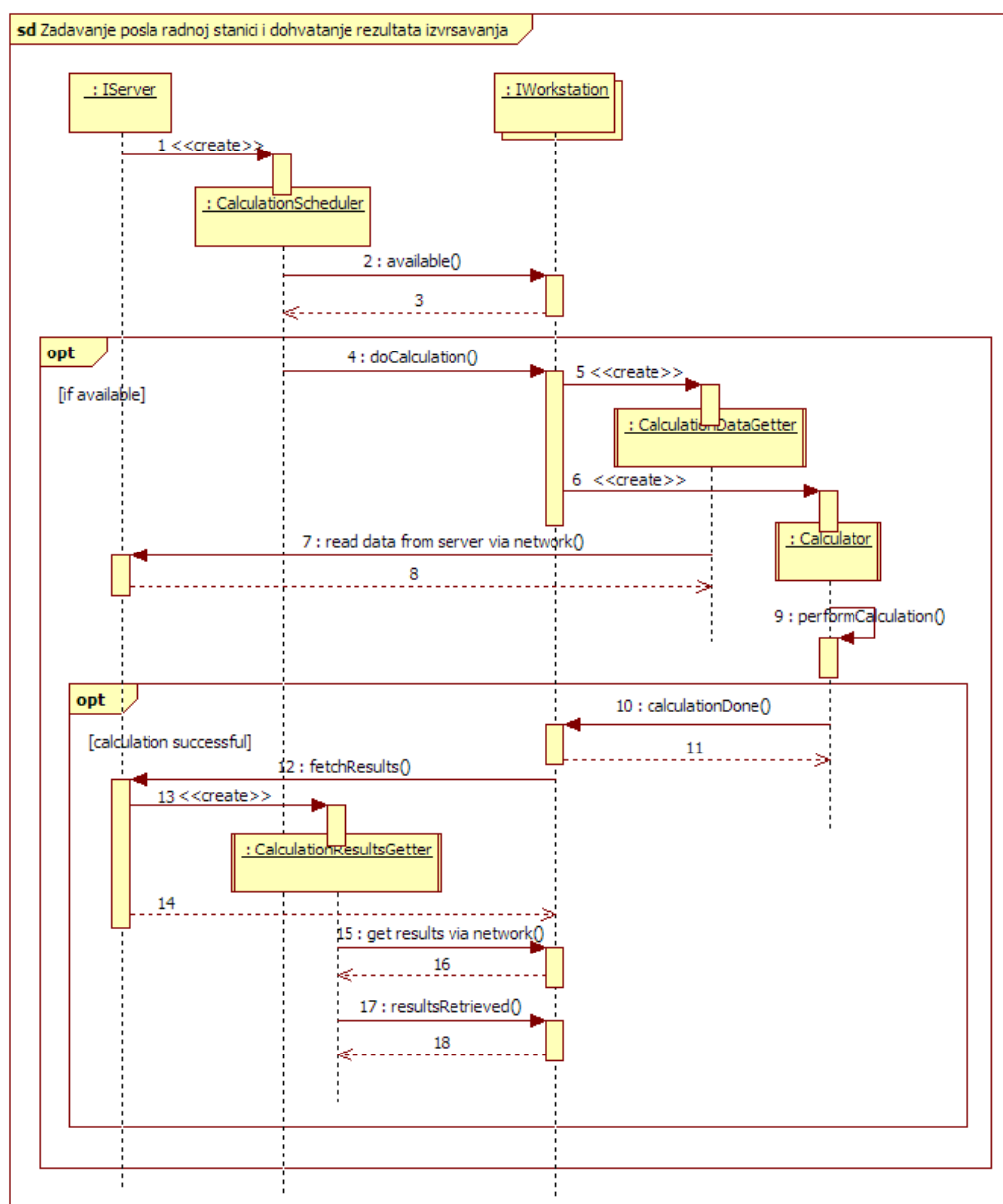
Дијаграм интеракције: Провера статуса посла
и дохватање резултата (клијент - сервер)



Дијаграм интеракције: Пријављивање радне станице
(радна станица - сервер)



Дијаграм интеракције: Одјављивање радне станице
(радна станица - сервер)



Дијаграм интеракције: Задавање посла радној станици и дохватање резултата (радна станица - сервер)

УПУТСТВО ЗА КОРИШЋЕЊЕ

Сервер се покреће задавањем команде *java -Djava.security.policy=java.policy Server.JFServer*. Радни директоријум сервера треба да садржи текстуални фајл под називом *server.config*. У њему се налазе параметри за конфигурисање сервера а то су: **број слободног порта** преко кога ће сервер примати захтеве од корисника и радних станица и **име** под којим ће сервер бити видљив клијентима и радним станицама. Сервер податке о пословима чува у бинарној датотеци *calcs.bin* као и у текстуалној датотеци *calcs.txt*. Све улазне податке о пословима као и резултате прикупљене од радних станица сервер чува у радном директоријуму.

Клијент се покреће задавањем команде *java -Djava.security.policy=java.policy Client.JFClient*. Радни директоријум клијента треба да садржи текстуални фајл *client.config*. У њему се налазе параметри за конфигурисање клијента, а то су:

- 1) IP адреса сервера;
- 2) број порта преко ког треба контактирати сервер;
- 3) име под којим је сервер објављен;
- 4) један слободан порт клијента преко кога ће га сервер контактирати;
- 5) име фајла у који клијент уписује информације о пословима које је тражио;

Уколико по стартовању клијента сервер није доступан клијент ће приказати информацију о томе и једино што ће у таквој ситуацији моћи да се уради је да покушај поновног повезивања на сервер са истим параметрима или измена параметара у конфигурационом фајлу и рестартовање програма.

Ако је веза са сервером успостављена корисник може да захтева извршење неког посла на два начина: Први је избором текстуалног фајла који описује захтевани посао. Тај фајл треба да има формат:

```
command=име команде
inMatrix=име улазне матрице
data=име улазне матрице
result=име излазног фајла
```

Име команде казује коју врсту посла треба одрадити и дозвољене вредности су: *solve*, *calculate*, *getLUdecomposition*, *getEigenvalueDecomposition*, *getInverseMatrix* и *getMatrixDeterminant*. Име улазне матрице је име фајла у коме се налази матрица над којом се врши израчунавање. Име излазног фајла је име фајла у коме ће се након успешног израчунавања налазити резултат. Поље *data* је опционо и оно је потребно само када је врста посла *solve* или *calculate*.

Други начин за слање захтева за извршење посла је да се параметри посла (*command*, *inMatrix*, *data* и *result*) изабери путем корисничког интерфејса.

После извесног времена од стартовања посла сервер ће обавестити клијента да су сви подаци потребни за израчунавање посла успешно ископирани и да је тиме посао прихваћен. Клијент кроз кориснички интерфејс може захтевати да добије статус у коме се посао тренутно налази. Када клијент добије информацију да је посао завршен (статус *READY*) он може захтевати резултате посла, чиме ће фајл који садржи резултате бити ископиран са сервера на клијент.

Радна станица се покреће задавањем команде *java -Djava.security.policy=java.policy Workstation.JFWorkstation*. Радни директоријум радне станице треба да садржи текстуални фајл *workstation.config*. У њему се налазе параметри за конфигурисање радне станице, а то су:

- 1) IP адреса сервера;
- 2) број порта преко ког треба контактирати сервер;
- 3) име под којим је сервер објављен;
- 4) порт за комуникацију радне станице и сервера;
- 5) име под којим је радна станица објављена;
- 6) један слободан порт радне станице преко кога ће га остале радне станице контактирати;
- 7) број нити које радна станица може да извршава

УПУТСТВО ЗА КОРИШЋЕЊЕ У ЦИЉУ ДАЉЕГ УСАВРШАВАЊА

Целокупно решење је урађено у Јави (верзија 1.6). Додатн, коришћен је пакет RMI помоћу кога је реализован удаљен позив процедура и пакет RMIIO за потербе читања датотека на удаљеним рачунарима. Пакет RMIIO се може наћи на сајту <http://openhms.sourceforge.net/rmio/>. Графички кориснички интерфејс је направљен помоћу пакета SWING.

Сви битни имплементациони детаљи су дати раније у тексту.

ПРИМЕР РАДА ПРОГРАМА У РЕГУЛАРНИМ И ВАНРЕДНИМ СИТУАЦИЈАМА

РЕГУЛАРНА СИТУАЦИЈА

Покрећу се сервер, клијенти и радне станице. Путем мреже се успоставља комуникација. Клијенти задају послове и гасе се регуларно. Сервер тражене послове распоређује напријављене радне станице, које се по завршетку обраде јављају серверу. Он купи резултате.

Клијенти касније траже резултате послова и добијају их уколико је посао успешно обављен. У супротном добијају поруку о тренутном стању посла.

ВАНРЕДНА СИТУАЦИЈА

Покрећу се сервер, клијенти и радне станице. Путем мреже се успоставља комуникација. Клијент задаје посао, међутим у току слања улазних фајлова веза између клијента и сервера се прекида (прекид због квара на мрежи, рачунар на коме је клијент је престао да ради, ...) Сервер обуставља све што је радио у вези са датим послом и брише све креиране записе и фајлове и наставља са радом као да захтева за тим послом није ни било.

ТЕСТОВИ ПЕРФОРМАНСИ

Перформансе су тестиране послом који представља решавање система од 2000 једначина са 2000 непознатих. Посао се обрађује око пола минута, независно од тога да ли је на располагању само једна или више радних станица (просечни РС рачунари).