2020.

Индустријски комуникациони протоколи у ЕЕ системима

ПРОЈЕКАТ А5

ПРЕДРАГ ГЛАВАШ & СТЕФАН БЕШОВИЋ

САДРЖАЈ

1. УВОД ... 3

1.1 Опис проблема ... 3

1.2 Циљеви задатка ... 3

2. ДИЗАЈН РЕШЕЊА ... 3

2.1 Дизајн серверске апликације ... 3

Контролна нит ... 4

Нит за слушање захтева ... 4

Нити обрађивачи ... 4

2.2 Дизајн клијентске апликације ... 5

Нит за комуникацију са сервером ... 5

Нити за иницирање комуникације са другим клијентима ... 6

Нит за слушање захтева клијената ... 6

Нити за обраду захтева клијената ... 6

2.3 Дизајн статичких библиотека ... 7

Connection Funcs ... 7

Data Structures ... 7

File IO Functions ... 8

Peer To Peer File Transfer Functions ... 8

Server Functions ... 8

ClientData ... 9

3. СТРУКТУРЕ ПОДАТАКА ... 11

3.1 Серверске структуре података ... 11

Табела утичница у обради ... 11

Табела информација о клијентима ... 11

Табела фајлова који се учитавају ... 11

Ред за чекање утичница на обраду ... 11

Табела информација о фајловима ... 12

Листа утичница повезаних клијената ... 12

3.2 Клијентске структуре података ... 12

Табела утичница у обради ... 12

Листа утичница повезаних клијената ... 13

Ред за чекање утичница на обраду ... 13

Ред за слање захтева клијентима ... 13

Листа чуваних делића ... 13

Структура за састављање фајла ... 13

3.3 Комуникационе структуре података ... 14

Захтев за фајл ... 14

Одговор на захтев за фајл 14
Захтев за део фајла 14

4. РЕЗУЛТАТИ ТЕСТИРАЊА АПЛИКАЦИЈЕ 14

4.1 Резултати првог серверског стрес теста 14

4.2 Резултати другог серверског стрес теста 16

5. ЗАКЉУЧАК ТЕСТОВА 18

5.1 Закључци првог серверског стрес теста 18

5.2 Закључци другог серверског стрес теста 18

6. ПОТЕНЦИЈАЛНА УНАПРЕЂЕЊА 19

6.1 Унапређења серверске апликације 19

1. Увод

1.1 Опис проблема

Потребно је направити клијент-сервер апликацију у C/C++ програмском језику која омогућава трансфер текстуалних и бинарних фајлова, арбитрарне величине, у клијент-сервер и „peer-to-peer“ режиму. Ово подразумева да клијент приликом скидања фајла неке делове може добити од сервера, а неке од других клијената који су тренутно активни и поседују делове тог фајла, уколико нема клијената са деловима онда се цео фајл преузима са сервера.

Сервер мора подржати чување и слање више фајлова клијентима који му се јаве захтевом, а клијент мора подржати слање дела једног или више фајлова другим клијентима који му се јаве захтевима. Клијентска апликација мора бити способна да паралелно тражи делове фајла који скида од више других клијената који те делове поседују.

И сервер и клијент морају подржати обрађивање више захтева истовремено употребом нити и њиховом адекватном синхронизацијом.

1.2 Циљеви задатка

Испунити све захтеве побројане у опису проблема уз коришћење критичних секција, семафора и нити. Пројектовати апликацију тако да се меморијски и процесорски ресурси употребљавају рационално, без коришћења уграђених структура података. Побринути се да су све методе издвојене у посебне .h и .cpp фајлове, да су документоване и да се налазе у посебним библиотекама. Извршити два стрес теста и документовати резултате који се добију. Резултати треба да докажу рационално управљање меморијским и процесорским ресурсима. Такође осигурати *Gracefully Shutdown* читаве апликације.

2. Дизајн решења

2.1 Дизајн серверске апликације

Серверска апликација се структурно може поделити на 3 групе нити:

1. Нити које обрађују корисничке захтеве
2. Нит која слуша нове захтеве за конекцију и прима захтеве од клијената
3. Контролна нит оператора сервера

Контролна нит

Контролна нит сервера је задужена да ослушкује команду за прекид рада сервера коју оператор издаје притиском на тастер „Q“ на тастатури, њена функционалност је једноставна и не ослања се на друге пројектне библиотеке.

Нит за слушање захтева

Нит која слуша захтеве ради са утичницама у неблокирајућем режиму мултиплексирањем операција заснованих на догађајима. У свом раду се ослања на *ConnectionFuncs* библиотеку о којој ће бити речи касније. Уколико се прими захтев за конекцију, а постоји простор за конекцију додатног клијента на серверу, извршиће пријем захтева, креирати утичницу за корисника и ставити је у листу утичница.

Уколико пристигну нови захтеви од већ конектованих клијента, утврђује се који клијенти су послали захтеве и затим се њихове утичнице стављају у ред за чекање који опслужују нити обрађивачи. Путем семафора се сигнализира нитима да су пристигли нови захтеви. Битно је напоменути да се утичнице на које су пристигли захтеви привремено уклањају из структуре за мултиплексирање догађаја како би се избегло вишеструко детектовање истих захтева на утичницама.

Уколико је немогуће прихватити нови захтев за конекцију клијента, из разлога који нису везани за ограничење максималног броја клијената, сервер ће се угасити и исписати грешку на конзолу.

Нити обрађивачи

Нити обрађивачи чине саму срж функционалности серверске апликације и у свом раду се првенствено ослањају на библиотеку *ServerFuncs,* али и на раније поменуту, *ConnectionFuncs* библиотеку. Број ових нити у апликацији се дефинише макроом *SERVER_THREADS.*

Ове нити након сигнала семафора скидају из реда за чекање утичницу и покушавају пријем захтева за фајл. Затим проверавају да ли је захтевани фајл већ учитан у меморију сервера, у зависности од тога можемо разликовати следеће случајеве:

1. Фајл је учитан у меморију
2. Фајл није учитан, али га тренутно учитава нека друга нит обрађивач
3. Фајл није учитан и тренутно га ниједна нит не учитава

Уколико је фајл учитан запаковаће одговор на клијентов захтев уз помоћ *ServerFuncs* библиотеке. Уколико фајл тренутно учитава нека друга нит, остале нити ће сачекати да она заврши учитавање, а затим запаковати одговор клијенту. Уколико фајл није учитан, нит ће уписати име фајла у табелу учитаваних фајлова и започети читање. Ако је читање неуспешно, обавестиће клијента да фајл не постоји, а ако је успешно поделиће фајл на делове и уписати податке о фајлу у табелу информација о фајловима.

Ако фајл постоји на серверу и постоје делови које има искључиво сервер, нити обрађивачи ће приступити слању делова клијенту.

У домену размене захетва, одговора и делова фајла и серверска и клијентска апликација се ослањају на *PeerToPeerFileTransferFunctions* библиотеку.

Битно је напоменути да се очекује да нити обрађивачи детектују и обраде клијенте који су раскинули везу са сервером и уклоне њихову утичницу из листе утичница.

2.2 Дизајн клијентске апликације

Клијентска апликација се може поделити на 4 групе нити:

1. Нит која комуницира искључиво са сервером и обрађује одговоре
2. Нити које иницирају комуникацију са другим клијентима и обрађују одговоре
3. Нит за слушање захтева клијената
4. Нити за обрађивање захтева клијената

Нит за комуникацију са сервером

Нит за комуникацију искључиво са сервером у свом раду се првенствено ослања на *ClientData* библиотеку, али, у циљу уредног затварања клијентске апликације, приликом грешке у комуникацији са сервером, ослања се и на *ConnectionFuncs* библиотеку.

Кориснику се омогућава да унесе назив жељеног фајла, помоћу којег се креира захтев за фајл-ом, који се шаље серверу. Приликом примања одговора од сервера, уколико фајл не постоји на серверу, кориснику се омогућава да опет тражи фајл од сервера, у супротном, примљени одговор даље наставља да се обрађује.
Уколико сервер садржи све делове фајла код себе, читав посао ће се одрадити у овој нити, где ће сервер да шаље све делове, један по један, и они ће тим редом да попуњава структуру у којој се фајл саставља.
Уколико сервер не садржи све делове фајла, већ су неки делови код других клијената, у овој нити ће се одрадити примање делова који се налазе на серверу, с тим што ће се пре тога активирати нити задужене за креирање и слање захтева за сваки од тих делова од клијената.
Ова нит чека да се скину сви делови фајла, где затим прави директоријум, чији је назив састављен од ИП и ПОРТ-а клијента и у њега се уписује скинути фајл.
Када се упис заврши, корисник има могућност поновног уноса назива фајла.

Битно је нагласити да уколико дође до испада сервера, ова нит се уништава уз комплетно гашење клијентске апликације.

Нити за иницирање комуникације са другим клијентима

Нит за комуникацију са другим клијентима у свом раду се првенствено ослања на *ClientData* библиотеку, али, у циљу уредног затварања утичница, ослања се и на *ConnectionFuncs* библиотеку. Број ових нити се у апликацији дефинише као макро *OUT_QUEUE_THREADS.*

Ове нити након сигнала семафора, скидају из реда за чекање структуру у којој се налазе потребне информације за креирање и слање захтева за делићем од клијента.

Након успешне конекције са клијентом, шаље се захтев за делићем. Када се захтев послао, нит очекује да прими одговор који ће садржати тражени делић.

Ако нит прими уредно делић, без грешака, структура у којој састављамо делиће ће се допунити за примљени делић и након тога нит је завршила са радом.

Уколико се деси грешка, нит ће одмах престати са радом.

Нит за слушање захтева клијената

Нит која слуша захтеве клијената ради са утичницама у неблокирајућем режиму мултиплексирањем операција заснованих на догађајима. У свом раду се ослања на *ConnectionFuncs* библиотеку о којој ће бити речи касније. Уколико се прими захтев за конекцију, извршиће пријем захетва, креирати утичницу за клијента и ставити је у листу утичница.

Уколико пристигну нови захтеви од већ конектованих клијента, утврђује се који клијенти су послали захтеве и затим се њихове утичнице стављају у ред за чекање, који опслужују нити за обраду захтева. Путем семафора се сигнализира нитима да су пристигли нови захтеви. Битно је напоменути да се утичнице на које су пристигли захтеви привремено уклањају из структуре за мултиплексирање догађаја како би се избегло вишеструко детектовање истих захтева на утичницама.

Нити за обраду захтева клијената

Нити за обраду захтева клијената у свом раду се првенствено ослањају на библиотеку *ServerFuncs,* али и на раније поменуту, *ConnectionFuncs* библиотеку. Број ових нити у апликацији се дефинише макроом *INC_QUEUE_THREADS.*

Ове нити након сигнала семафора, скидају из реда за чекање утичницу и покушавају пријем захтева за делићем. Затим проверавају да ли делић постоји на клијенту у листи сачуваних делића. Уколико делић постоји, као одговор се шаље жељени делић клијенту, у супротном, као одговор се шаље измењена структура, која чува податке о делићу, тако да клијент, када прими одговор, зна да делић није на клијенту.

Као што је описано у делу 2.1. који говори о дизајну серверске апликације, обе апликације се ослањају на *PeerToPeerFileTransferFunctions* библиотеку.

Такође, у клијентској апликацији се очекује да нити за обраду захтева клијената детектују и обраде клијенте који су раскинули везу и поступе у одговарајућем маниру.

2.3 Дизајн статичких библиотека

У оквиру пројекта постоји више статичких библиотека које садрже функционалности које користе клијентска и серверска апликација или су им заједничке. Пројектне библиотеке су следеће:

1. ConnectionFuncs
2. DataStructures
3. FileIO_Functions
4. PeerToPeerFileTransferFunctions
5. SeverFunctions
6. ClientData

Следи опис функционалности библиотека. Детаљан опис посла који ради свака конкретна функција у некој библиотеци, списак њених аргумената и повратна вредност се могу наћи у одговарајућим .h фајловима библиотека.

Connection Funcs

Ова библиотека садржи функције везане за управљање конекцијама и утичницама, прихватање захтева за повезивање, детекцију утичница на које су пристигли захтеви, детекцију затворених/покварених утичница и управљање листом утичница. Сама библиотека је заштићена критичном секцијом и самим тиме се може користити у апликацијама које имају више нити.

Пре употребе библиотеке је потребно позвати иницијализатор њеног управљача како би се иницијализовала критична секција, а након употребе је потребно позвати брисање управљача.

Data Structures

Овде се налазе имлементације структура података које се користе у пројекту, а то су:

1. Ред за чекање
2. Двоструко повезана листа
3. Хеш мапа

Ред за чекање је имплементиран тако да увек буде исте дужине, оне која се проследи његовом конструктору. Безбедан је за коришћење од стране више нити и може чувати произвољне типове података.

Двоструко повезана листа динамички проширива и може бити произвољне дужине у зависности од броја додатих елемената, безбедна је за коришћење од стране више нити осим у домену претраге где се очекује од конзумента листе да део кода који претражује листу обезбеди мутексима/критичним секцијама.

Хеш мапа омогућава мапирање стринга као кључа на произвољан тип податка. Као и ред за чекање, увек је фиксне величине која се прослеђује кроз конструктор, али подржава неограничено уланчавање прекорачилаца у случају колизије кључева. Безбедна је за коришћење од стране више нити.

File IO Functions

Ова библиотека садржи функције везане за упис и читање из фајлова, понашање функција је недефинисано уколико више нити или процеса покуша да чита или пише један фајл.

Peer To Peer File Transfer Functions

Овде се налазе функције које чине саму срж комуникационог протокола у апликацији, као и структуре података које се размењују између клијената и сервера. Дефинисана макроима ту се налазе и ограничења система у погледу максималног броја клијената, дужине назива фајла и слично.

Све функције подржавају рад са утичницама које су у блокирајућем и неблокирајућем режиму и уколико је бафер утичнице претрпан сачекаће да се ослободи како би подаци били успешно примљени или послати. Функције за слање и пријем делова фајла подржавају слање фајлова произвољне дужине.

Битно је напоменти да овде важе ограничења која и иначе важе при раду са утичницама, те није могуће у исто време читати нити писати са утичнице из више нити . Корисницима библиотеке је остављено да се побрину на нивоу своје апликације да се ова ограничења испоштују.

Server Functions

Све функције за управљање серверским подацима (осим за управљање утичницама) се налазе у овој библиотеци. Подржано је динамичко управљање подацима корисника као и подацима о самим фајловима и деловима фајла.

Могуће је доделити клијентима делове произвољног броја фајлова које затраже, међутим подаци се чувају само током једне комуникационе сесије те ће бити трајно обрисани чим се клијент први пут одјави или прекине везу са сервером.

Сви фајлови се увек деле на фиксан број делова, а клијенту се додељује на чување било који део који је искључиво у поседу сервера. Уколико су сви делови додељени на чување клијентима, наредни клијент који се повеже добиће на чување дупликат једног од делова.

Приликом враћања дела фајла у посед сервера, због одјаве клијента или пуцања везе са њим, поукшаће се додела тог дела неком клијенту који има његов дупликат, уколико такви не постоје сервер ће преузети обавезу слања тог дела новим клијентима који га затраже.

Све функције безбедне су за коришћење од стране више нити.

Пре употребе ове библиотеке потребно је позвати иницијализатор њеног управљача како би се иницијализовала критична секције. Када се заврши са употребом потребно је позвати брисање управљача.

ClientData

У овом делу се налазе све функције које су потребне клијенту за рад са делићима који пристигну од сервера и од других клијената, као и за упис фајлова на диск. Подржано је динамичко управљање подацима о фајловима и деловима фајлова. У њој се такође налазе и структуре података које клијент користи уз структуре које су дефинисане у *Peer To Peer File Transfer Functions* библиотеци.

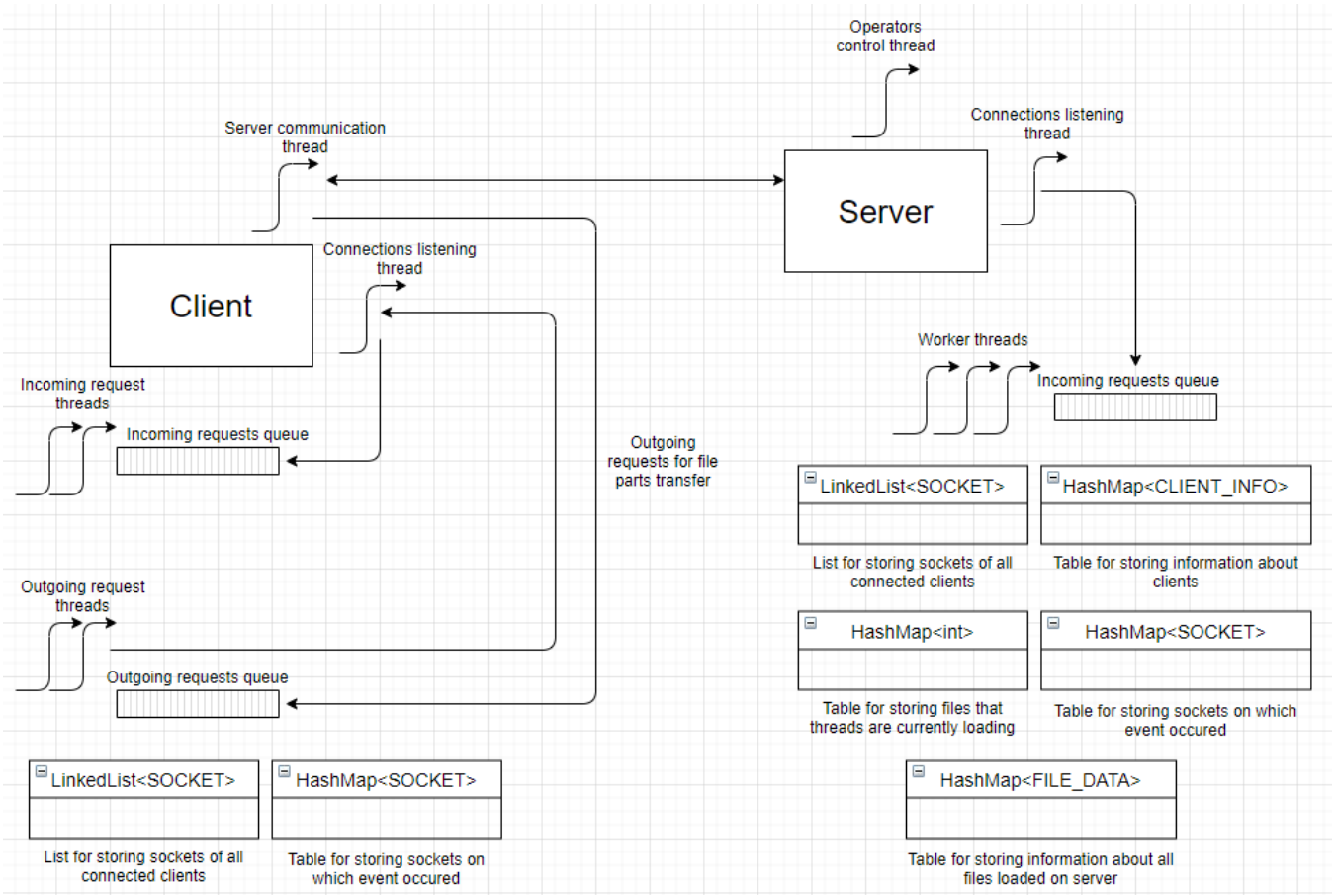
Сви делићи фајлова, које сервер означи да клијент треба да чува, се чувају у двоструко повезаној листи.

Сви пристигли делићи једног фајла се смештају у помоћну структуру у којој се делићи спајају и на крају исписују у фајл.

Фајл се исписује.

Приликом исписивања, помоћна структура у којој састављамо фајл се поново иницијализује и спремна је за поновну употребу.

Пре употребе ове библиотеке потребно је позвати два иницијализатора њених управљача како би се иницијализовале критичне секције. Када се заврши са употребом потребно је позвати брисање управљача.



1. Slika dijagrama dizajna aplikacije

3. Структуре података

3.1 Серверске структуре података

Сервер за обраду и привремено чување података користи следеће структуре:

1. Табелу утичница у обради
2. Табелу информација о фајловима
3. Табелу фајлова који се тренутно учитавају
4. Ред за чекање утичница за обраду
5. Табелу информација о клијентима
6. Листу утичница повезаних клијената

Табела утичница у обради

Ово је хеш мапа у коју се стављају утичнице за које је детектовано да се десио неки догађај на њима, утичнице остају у мапи док год једна од нити обрађивача не заврши обраду догађаја. Утичнице које су у овој табели се игноришу током мултиплексирања нових догађаја функцијом *select()* .

Табела информација о клијентима

То је хеш мапа која мапира стринг репрезентацију утичнице клијента на податке о њему релевантне за сервер. Ти подаци су:

- Утичница клијента
- Адреса утичнице на којој клијент слуша захтеве за повезивање
- Низ назива фајлова које је клијент скинуо са сервера
- Број фајлова које је клијент скинуо
- Дужина низа назива фајлова (може бити већа броја фајлова које је скинуо)

Хеш мапа је коришћена због неопходности брзог приступа подацима и претраге по кључу.

Табела фајлова који се учитавају

Табела фајлова који се учитавају је хеш мапа у којој се налазе називи фајлова које тренутно учитавају нити обрађивачи. Хеш мапа је искоришћена због потребе да се брзо провери да ли се назив фајла за који је пристигао захтев већ ту налази. Нити обрађивачи ће чекати са обрадом захтева док год је назив захтеваног фајла у овој табели.

Ред за чекање утичница на обраду

Ред за чекање утичница на обраду је ред у који се стављају утичнице за које је утврђено да се на њима десио догађај, нити обрађивачи узимају утичнице са почетка реда и хронолошки их обрађују.

Табела информација о фајловима

Табела информација о фајловима је хеш мапа која за кључ има назив фајла који се мапира на структуру података која описује сваки учитани фајл на серверу. Подаци о фајлу су:

- Показивач на бафер фајла
- Назив фајла
- Низ делова фајла
- Величина фајла
- Дужина низа делова
- Број делова додељених клијентима на чување

Подаци о сваком делу фајла су следећи:

- Број дела
- Показивач на почетак бафера дела
- Да ли је у власништву сервера
- Дужина дела
- Адреса утичнице клијента који поседује део (узима се у обзир само ако је клијент власник)

Листа утичница повезаних клијената

Листа утичница повезаних клијената је двоструко повезана листа у којој се чувају утичнице свих тренутно повезаних клијената. Листа је изабрана због честе потребе да се итерира кроз листу и додају и уклањају елементи.

3.2 Клијентске структуре података

Клијент за обраду и привремено чување података користи следеће структуре:

1. Табела утичница у обради
2. Листу утичница повезаних клијената
3. Ред за чекање утичница за обраду
4. Ред за слање захтева клијентима
5. Листа чуваних делића
6. Структура за састављање фајла

Табела утичница у обради

Ово је хеш мапа у коју се стављају утичнице за које је детектовано да се десио неки догађај на њима, утичнице остају у мапи док год једна од нити за обраду захтева клијената не заврши обраду догађаја. Утичнице које су у овој табели се игноришу током мултиплексирања нових догађаја функцијом *select()* .

Листа утичница повезаних клијената

Листа утичница повезаних клијената је двоструко повезана листа у којој се чувају утичнице свих тренутно повезаних клијената. Листа је изабрана због честе потребе да се итерира кроз листу и додају и уклањају елементи.

Ред за чекање утичница на обраду

Ред за чекање утичница на обраду је ред у који се стављају утичнице за које је утврђено да се на њима десио догађај, нити за обраду захтева клијената узимају утичнице са почетка реда и хронолошки их обрађују.

Ред за слање захтева клијентима

Ред у који се смешта структура података која садржи потребне информације за формирање и слање захтева за делићем који се налази код других клијената.
Структура садржи:

- Назив фајла
- Информације о делићу - садржане у структури у облику:
    - Адреса клијента
    - Број делића који чува

Листа чуваних делића

Листа за чување података о свим делићима фајлова, које је сервер доделио клијенту, ради њиховог прослеђивања другим клијентима.
Подаци о сваком делу:

- Назив фајла
- Показивач на почетак бафера дела
- Дужина дела

Листа се користи ради честе потребе да се кроз њу итерира и додају елементи.

Структура за састављање фајла

Структура података која се користи за састављање фајла од примљених делића.
Структура садржи следеће:

- Назив фајла
- Показивач на почетак бафера фајла
- Број скинутих делова
- Величину фајла
- Индекс делића за чување

3.3 Комуникационе структуре података

Структуре података које се размењују између сервера и клијената су:

- Захтев за фајл
- Одговор на захтев за фајл
- Захтев за део фајла
- Подаци о делу фајла

Захтев за фајл

Захтев за фајл чине назив фајла и адреса утичнице клијента који је послао захтев.

Одговор на захтев за фајл

Одговор на захтев за фајл чине следећи подаци:

- Да ли фајл постоји
- Број делова које имају клијенти
- Број делова које има сервер
- Број дела који је додељен клијенту на чување
- Величина фајла у бајтима
- Низ података о деловима које имају клијенти (узима се у обзир само ако има делова додељених клијентима)

Захтев за део фајла

Захтев за део фајла чине следећи подаци:

- Назив фајла
- Број дела

4. Резултати тестирања апликације

Над пројектом су извршена 2 стрес теста, тестирајући како серверска апликација функционише под оптерећењем. Резултати су у наставку.

4.1 Резултати првог серверског стрес теста

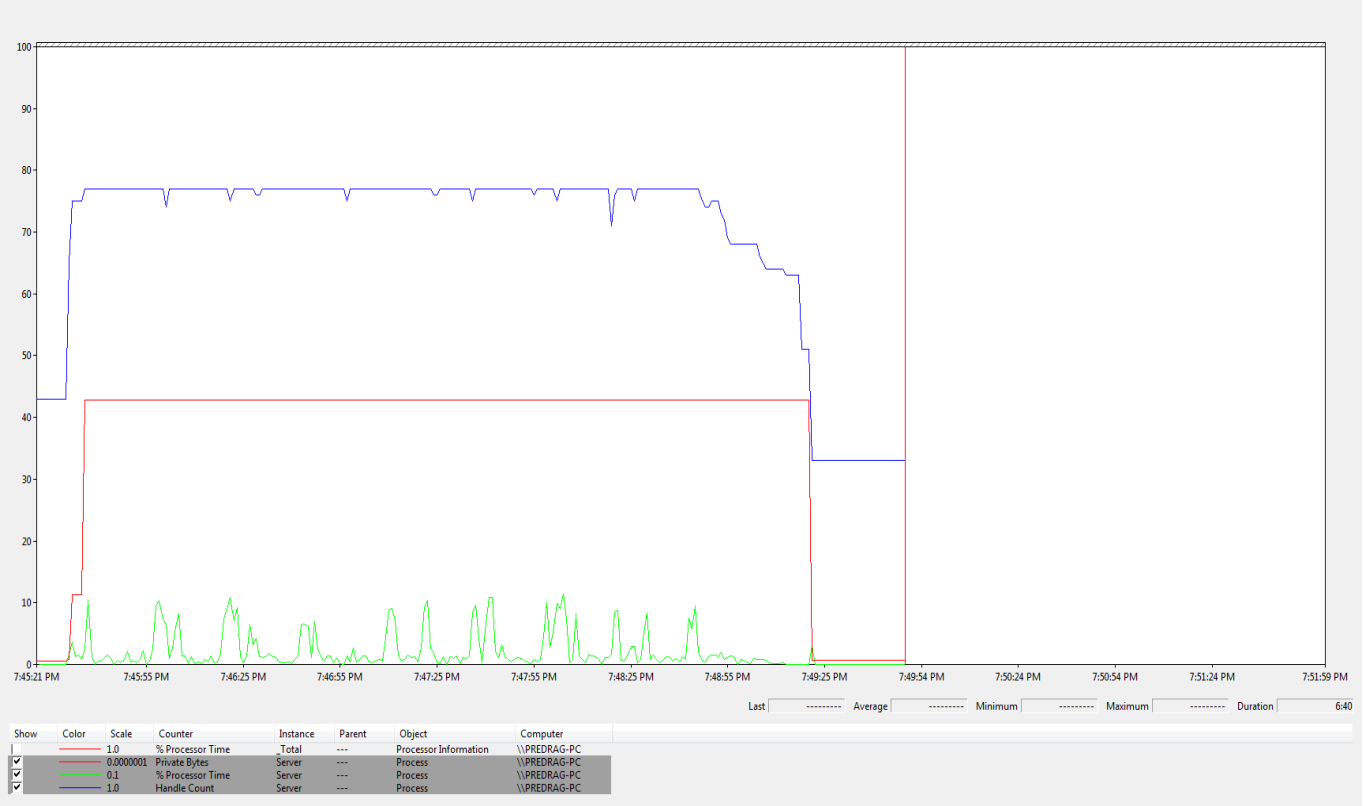
Први серверски стрес тест је направљен да симулира захтеве за фајловима разних величина од стране великог броја клијената у исто време.

У сврху тестирања су креирани фајлови од 100 КБ, 1 МБ, 100МБ и 300 МБ. У фајловима се налазе насумични карактери, а сами фајлови су направљени да буду тешки за компресију. Сервер симулираним клијентима шаље делове фајла, а они их након пријема одбацују.

Клијенте симулира 30 нити на стрес тест апликацији, при чему свака нит шаље 20 захтева за фајл при том се сваки пут повезујући на сервер изнова и представљајући се као други клијент. Између слања захтева није прављена пауза како би се адекватно симулирало оптерећење.

ID	Time	Allocations (Diff)	Heap Size (Diff)
1	3.53s	559 (n/a)	168.04 KB (n/a)
2	13.48s	642 (+83 ↑)	103,700.32 KB (+103,532.28 KB ↑)
3	48.40s	653 (+11 ↑)	103,705.74 KB (+5.42 KB ↑)
4	54.18s	651 (-2 ↓)	103,702.83 KB (-2.91 KB ↓)
5	60.90s	639 (-12 ↓)	103,701.89 KB (-0.94 KB ↓)
6	79.12s	620 (-19 ↓)	103,699.32 KB (-2.57 KB ↓)
7	91.52s	496 (-124 ↓)	133.55 KB (-103,565.77 KB ↓)

*2. Слика **heap**-а током теста*



3. Дијаграм перформанси током првог стрес тестирања сервера

На слици 3. изнад се налази дијаграм перформанси сервера током трајања стрес теста. Зеленом бојом је означен проценат укупног процесорског времена које је апликација користила, црвеном количину коришћене меморије у десетинама мегабајта, а плавом број руковалаца (хендлова) које сервер користи.

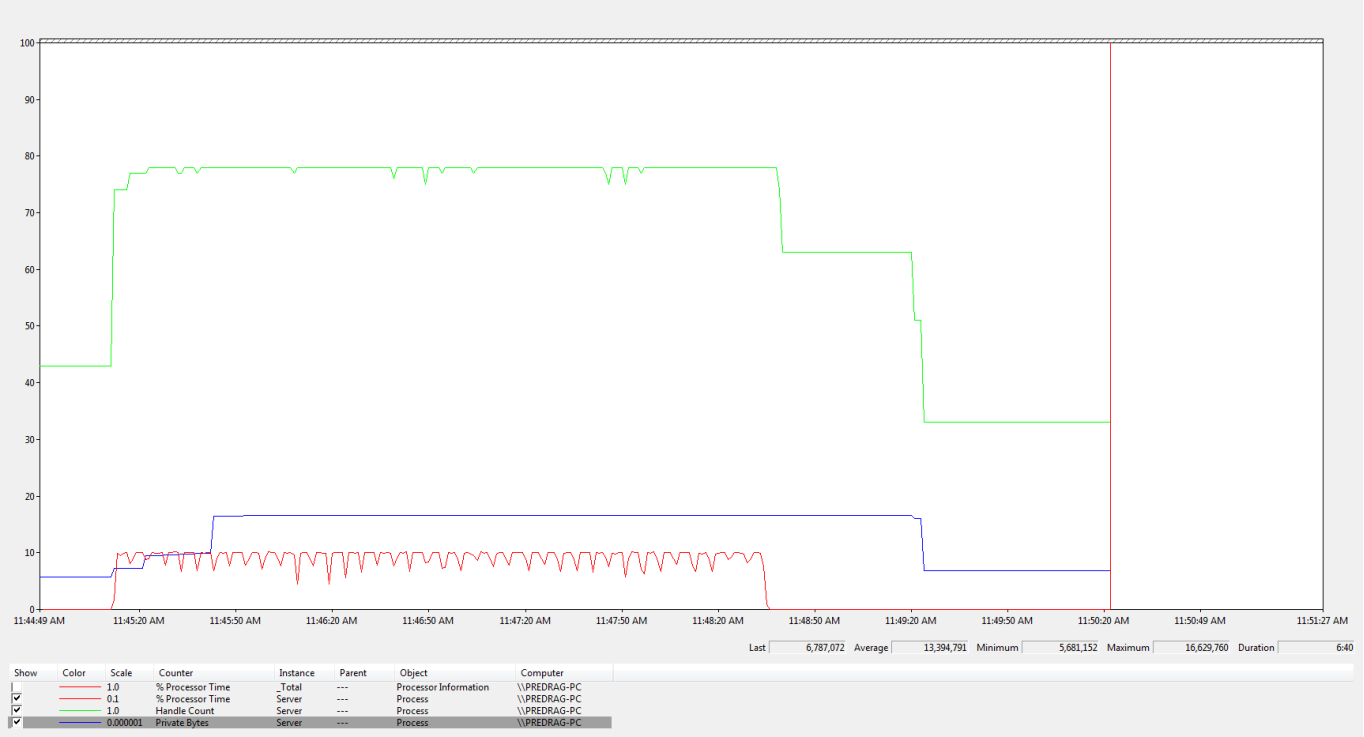
Из приложеног се види да нема цурења меморије нити руковалаца у апликацији.

4.2 Резултати другог серверског стрес теста

Други серверски стрес тест је направљен да симулира захтеве за фајловима мањих величина од стране великог броја клијената уз одржавање везе са сервером 5 секунди након завршетка пријема делова. Циљ извршавања овог теста је утврдити како се сервер понаша када велики број клијената остане да одржава везу са њим након што заврше пријем делова.

У сврху тестирања су креирани фајлови од 112 КБ, 326 КБ, 2.06МБ и 6.18 МБ. У фајловима се налазе ASCII карактери. Сервер симулираним клијентима шаље делове фајла, а они их након пријема одбацују.

Клијенте симулира 30 нити на стрес тест апликацији, при чему свака нит шаље 20 захтева за фајл при том се сваки пут повезујући на сервер изнова и представљајући се као други клијент. Између слања захтева је прављена пауза од 5 секунди како би се додатно онемогућио сервер да брзо обради надолазеће захтеве.



4. Дијаграм перформанси током другог стрес тестирања сервера

На слици 4. изнад се налази дијаграм перформанси сервера током трајања другог стрес теста. Зеленом бојом је означен број руковалаца које је апликација користила, плавом количина коришћене меморије у десетинама мегабајта, а црвеном проценат процесорског времена које сервер користи.

Из приложеног се види да нема цурења меморије нити руковалаца у апликацији.

5. Закључак тестова

5.1 Закључци првог серверског стрес теста

Из резултата првог серверског стрес теста се може закључити да на серверу нема цурења меморије нити руковалаца. Потрошња меморије је очекивана и износи приближно 400МБ, јер је то сума величина фајлова које су клијенти тражили, а сервер учитао у оперативну меморију. Мале флуктуације у потрошњи меморије постоје, јер се клијенти стално прикључују и искључују, па се стално уклањају и додају информације о њима, али се те флуктуације не виде на дијаграму, јер су реда величине пар килобајта.

Посматрањем искоришћености процесорског времена може се видети да постоје периоди изузетно слабе искоришћености процесора праћени наглим скоковима. То се делом може објаснити тиме што се током стрес теста дешава да на сервер пристигну захтеви за исти фајл у исто време од стране више клијената, то приморава остале нити да чекају на учитавање фајла пре него што могу наставити са обрадом. Друго објашњење слабе искоришћености серверског времена јесте величина критичних секција унутар серверске библиотеке са функцијама, као и чињеница да је приступ табели података ограничен на једну нит у једном моменту. Ово значи да се приликом обраде захтева, а конкретно при паковању одговора на клијентски захтев, ствара уско грло, јер тој функционалности може приступити искључиво једна нит у једном моменту.

5.2 Закључци другог серверског стрес теста

Из резултата другог серверског стрес теста се може закључити да на серверу нема цурења меморије нити руковалаца. Потрошња меморије износи приближно 20МБ уз мање скокове у првих 30 секунди који су последица учитавања фајлова у оперативну меморију. Ни на овом дијаграму се не могу уочити скокови и падови у потрошњи меморије услед пријаве и одјаве клијената, опет из разлога величине њиховим података у односу на величину фајлова у оперативној меморији .

Посматрањем искоришћености процесорског времена може се видети да је она изузетно мала, чак мања и у односу на први стрес тест. То се може објаснити тиме што је серверска листа клијената скоро стално пуна те нови клијенти који желе да се повежу и затраже фајл то не могу учинити док се неко од постојећих не одјави. Током тог периода сервер има минималну активност која се своди на мултиплексирање догађаја на утичницама и чекање.

6. Потенцијална унапређења

6.1 Унапређења серверске апликације

Из закључака теста серверске апликације јасно се види да је уско грло система библиотека са функционалностима за управљање подацима о фајловима, конкретно сваки приступ табели са фајловима је у принципу секвенцијалан, јер се ограничава број нити које му могу приступити на 1.

Један начин да се овај проблем превазиђе је увођење критичних секција на нивоу сваке структуре која чува податке о учитаном фајлу уместо досадашње критичне секције на нивоу целе табеле. Међутим ово потенцијално доводи до великог пораста броја употребљених руковалаца и додатне сложене логике за њихово адекватно иницијализовање и ослобађање након коришћења.

Друго решење би било увођење табеле обрађиваних фајлова, која би, по узору на табелу учитаваних фајлова, имала називе фајлова чији се подаци тренутно обрађују од стране неке од нити. Приступ табели се може обезбедити критичном секцијом која би била пар линија кода дуга. Ако се назив фајла налази у табели онда нити, које треба да му приступе, чекају на његово уклањање из табеле, некон чега приступају његовим подацима и стављају га опет у исту табели како би сигнализирале осталим нитима да сачекају по потреби.

Из другог теста се може извући закључак да је уско грло серверске апликације и ограничење максималног броја клијената на 15 постављено у апликацији. Ово је потпуно очекивано, а начин за решење овог проблема би био прављење листе делова фајла која би била динамички проширива и уклањање ограничења броја клијената из апликације.