Институт за математику и информатику
Природно-математички факултет
Универзитет у Крагујевцу

Семинарски рад из предмета Студијски истраживачки рад

Тестирање перформанси NFS и Lustre фајл система

Студент:
Ненад Ацковић

Професор:
др Милош Ивановић

Септембар 2015.

Садржај

Скраћенице **2**

1 *Увод* **3**

2 Тестирање брзине извршавања У/И операција **4**
2.1 Хардверска конфигурација кластера 4
2.2 Iozone тестирање 5
2.2.1 Инсталација и покретање програма 6
2.3 Тестирање помоћу `dd` програма 7
2.4 Game of Life 10
2.4.1 Правила 10
2.4.2 Порекло 10
2.5 Опис програма 11
2.6 Резултати тестирања 13

3 Закључак **18**

Скраћенице

HPC High-Performance Computing. 2

MPI Message-Passing Interface. 2

NFS Network File Sistem. 4

Глава 1

Увод

Овај рад као примарни циљ поставља проучавање могућности *Lustre* фајл система у унапређењу рада кластера високих перформанси, као и користи од паралелних улазно излазних операција дефинисаних MPI-2 стандардом. *Lustre* паралелни фајл систем има предност над осталим (секвенцијалним) системима услед могућности обављања улазно-излазних операција у паралелном маниру. Са друге стране, MPI-2 представља индустријски стандард за развој паралелних програма, а његове имплементације подржавају паралелно читање и писање на уређаје масовне меморије. У циљу сагледавања начина да се побољшају укупне перформансе локалног кластера *Medflow*, упоређују се брзине читања и писања података у фајлове који се налазе како на *Lustre*, тако и на NFS фајл систему.

Кластер високих перформанси (*HPC*) представља скуп рачунара умрежених коришћењем локалне мрежне инфраструктуре, помоћу које међусобно комуницирају. Коришћење специфичне програмске подршке даје висок степен интеграције рачунара, омогућава њихов координирани заједнички рад и претвара их ефективно у јединствен вишепроцесорски систем који користи дистрибуирану меморију.

За истраживачке радове који захтевају обимне математичке прорачуне данас се углавном користе паралелни програми који на кластеру високих перформанси рашчлањују проблем на више рачунарских процесора. Тиме се време добијања резултата знатно смањује у односу на време потребно за добијање резултата на једном процесору, понекад за више редова величине. Најефективнији начин је употреба поменутог MPI стандарда за развој паралелних програма. На Универзитету у Крагујевцу постоји неколико HPC кластера, али ни један од њих не поседује паралелни фајл систем. Та чињеница у појединим случајевима употребе може да доведе до озбиљног пада перформанси. На срећу, први кластер са паралелним фајл системом постављен је недавно у истраживачко развојном центру за биоинжеering у Крагујевцу. Након иницијалне инсталације HPC кластера, дошло се на идеју да се истраже реалне могућности и употребна вредност новоинсталираног *Lustre* система, што и представља основну мотивацију за израду овог рада.

У раду су дата тестирања брзина извршавања операција фајл система помоћу програма *Iozone*, *dd* и *Game of Life*. Под претпоставком да постоји разлика у брзини улазно/излазних операција ових фајл система, анализирају се добијени резултати тестирања и на основу њих се утврђује који је погоднији за који домен употребе.

Глава 2

Тестирање брзине извршавања У/И операција

Како би се увидела предност *Lustre* фајл система у односу на NFS фајл систем, вршено је мерење брзине извршавања У/И операција на *Medflow* кластеру високих перформанси који користи оба наведена фајл система. Тестирање је вршено помоћу три различите апликације:

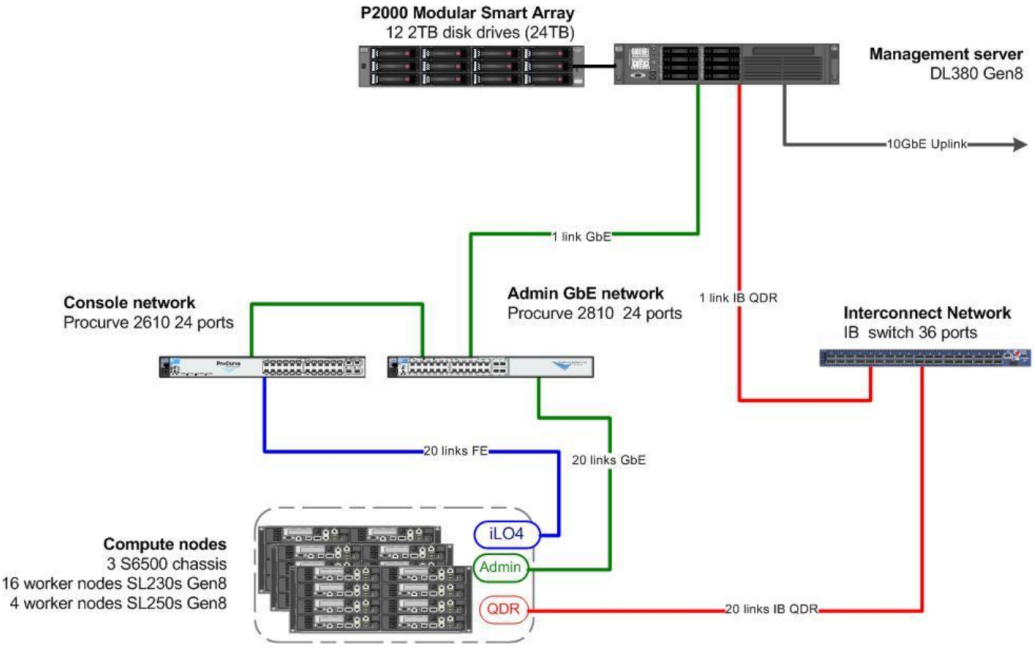
- *Iozone* - апликација намењена тестирању перформанси улазно-излазних операција,
- `dd` - једноставан UNIX системски алат за конверзију и копирање фајлова,
- *Game of Life* - "реална" апликација која поред У/И операција поседује и делове у којима се интензивно рачуна.

2.1 Хардверска конфигурација кластера

Medflow кластер је састављен од HP Proliant SL230s Gen8 и HP Proliant SL250s Gen8 радних чворова, заснованих на Intel® Xeon® E5-2600 (Sandy Bridge) процесорима. Кластер се састоји од следећих компоненти:

- 18 радних чворова HP Proliant SL230s Gen8 са 2 Intel® Xeon® E5-2660 процесора (2.2GHz – 8 cores - 20MB L3 cache - 95W) и 64GB RAM меморије.
- 4 радна чвора P Proliant SL250s са 2 Intel® Xeon® E5-2670 (2.6GHz – 8 cores - 20MB L3 cache - 115W) процесора и 64GB RAM меморије. Сваки од ових чворова садржи и 1 GPU NVIDIA Tesla M2090 6G графичку картицу.
- Једног *Mellanox Infiniband* QDR свич преко кога су повезани чворови.
- Једног управљачког сервера DL380pGen8 који омогућава пријаву на систем.
- Једног управљачког сервера DL380pGen8 који омогућава функционисање паралелног фајл система.
- Система за складиштење података који садржи 1 *HP P2000 G3 Modular Smart Array System* повезан на управљачки сервер, што даје укупно 12TB простора (6 HDD дискова по 2TB). Низ је конфигурисан као RAID ниво 6. Сви чворови могу приступити овим дисковима помоћу *Lustre* протокола.

Кластер је конфигурисан са Linux Kernel 2.6 на свим чворовима. Scientific Linux 6.6 64-bit је инсталиран како на управљачком чвору, тако и на радним чворовима. Ресурсима кластера се управља помоћу *TORQUE* и *Maui* сервиса. Први је намењен контроли и управљању кластер пословима, док је други распоређивач.



Слика 2.1: Компоненте кластера

Програмско окружење је засновано на OpenMPI библиотекама за Linux оперативни систем. На кластеру је инсталирана OpenMPI верзија 1.6.5. Треба нагласити и да је NFS фајл систем инсталиран постављен само на једном диску, док Lustre користи цео RAID6 низ.

2.2 Iozone тестирање

Iozone је алат за мерење брзине У/И операција фајл система. Он генерише и мери трајање великог броја операција са фајловима. *Iozone* може бити инсталиран на великом броју архитектура и такође може радити у оквиру многих оперативних система. Тестирање се врши кроз следеће операције:

- *Write* - мери брзину уписа података у нови фајл.
- *Re-write* - мери брзину уписа у фајл који већ постоји
- *Read* - мери брзину читања из фајла
- *Re-read* - мери брзину читања из фајла који је претходно прочитан
- *Random read* - мери брзину читања из фајла са насумичним приступом локацијама унутар фајла
- *Random write* - мери брзину писања у фајл са насумичним приступом локацијама унутар фајла
- *Random mix* - мери брзину писања и читања из фајла са насумичним приступом локацијама унутар фајла
- *Backwards read* - мери брзину читања из фајла уназад

- *Record rewrite* - мери брзину писања података у одређени део фајла
- *Strided read* - мери брзину читања из фајла са тачно одређеним параметрима
- *Fwrite* - мери брзину писања у фајл помоћу функције `fwrite()`
- *Fread* - мери брзину читања из фајла помоћу функције `fread()`
- *Freread* - мери поновно читања из фајла помоћу функције `fwrite()`

2.2.1 Инсталација и покретање програма

Iozone програм се инсталира помоћу следећих команди:

Листинг 2.1: Инсталација Iozone

```
wget http://www.iozone.org/src/current/iozone3_394.tar
tar xvf iozone3_394.tar
cd iozone3_394/src/current
make
make linux
```

Овај програм је могуће покренути и помоћу великог број параметара. Конкретни параметри који су коришћени за тестирање на кластеру имају следеће значење:

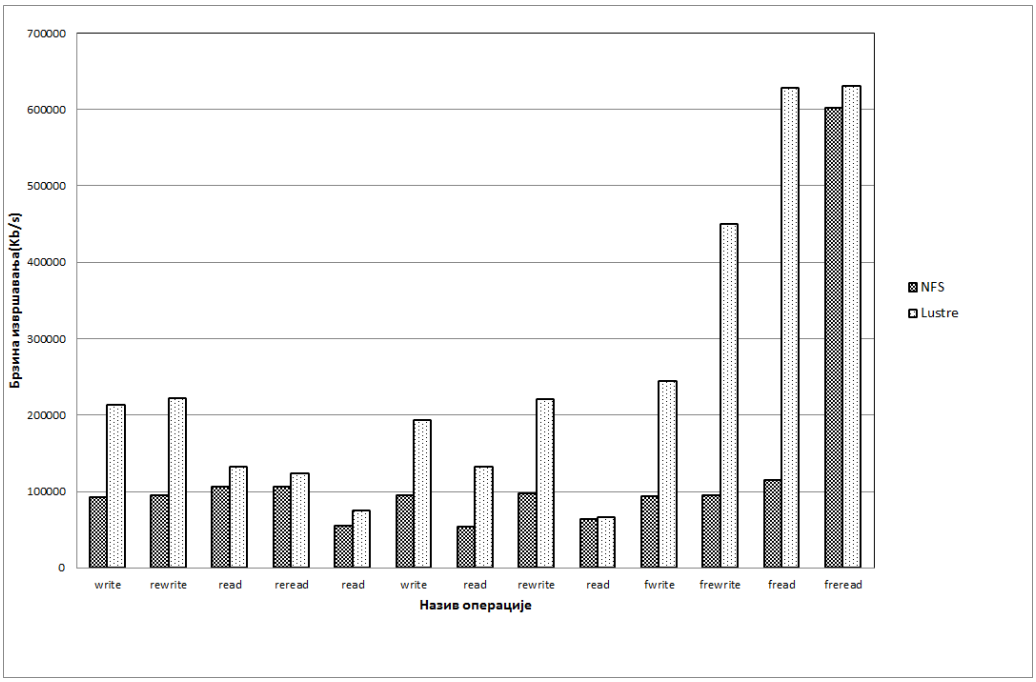
- `-b назив фајла` - генерисање Excel излазног фајла
- `-c` - мери и време које је потребно за функцију *close()*
- `-I` - користи *DIRECT I/O* заставицу за све фајл операције
- `-o` - уписује синхроно на диск. *Iozone* отвара фајл са *O_SYNC* заставицом
- `-r` - одређује величину записа у килобајтима
- `-s` - одређује величину фајла који се користи за тестирање

Програм се покреће помоћу команде:

```
./iozone -s 25000000 -r 1024 -I -c -o -b output.xls
```

Резултати програма се могу видети у фајлу *output.xls*.

Посматрајући резултате представљене на Слици 4.2, закључујемо да је брзина извршавања улазно/излазних операција *Lustre* фајл система значајно већа у односу на брзину NFS система без обзира на врсту операције која се извршава. Највећа разлика у брзини се уочава код теста који користи `fread()` команду, док је најмања разлика у брзини при извршавању операције обичног читања.



Слика 2.2: *Iozone* резултати

2.3 Тестирање помоћу `dd` програма

`dd` је једноставан алат који служи за писање и читање блокова података диска. Он такође мери и брзину којом је операција извршена.

Параметри командне линије су:

- `if` - улазни фајл
- `of` - излазни фајл
- `bs` - величина блока
- `count` - број блокова

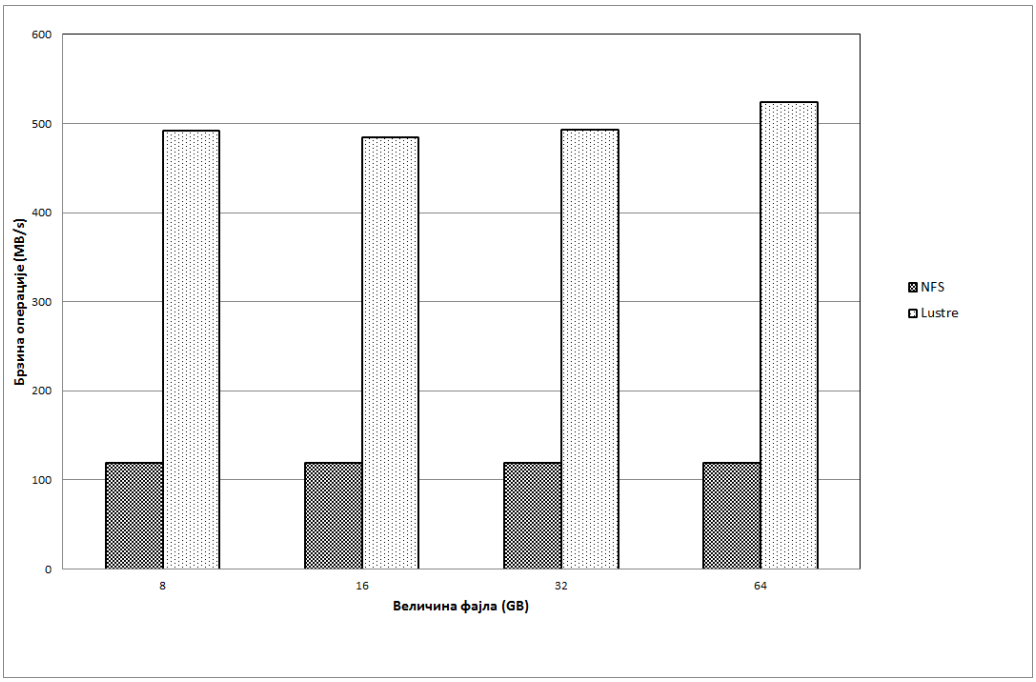
Програм за упис у фајл се покреће командом:

```
dd if=/dev/zero bs=1M count=16384 of=file_16GB
```

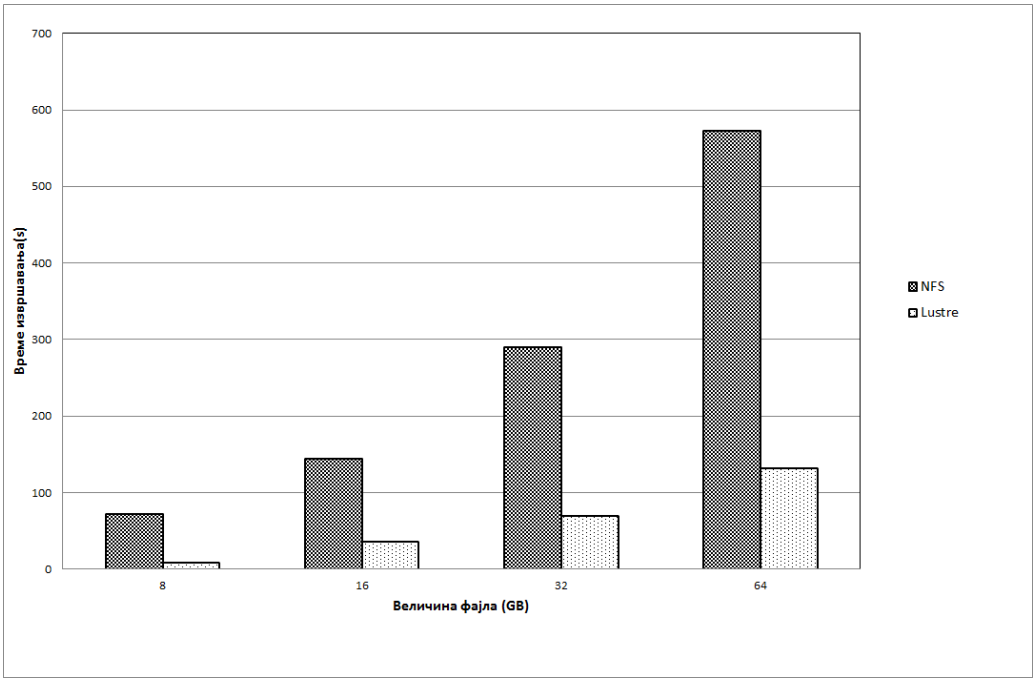
а за читање из фајла:

```
dd if=file_16GB bs=1M of=/dev/null
```

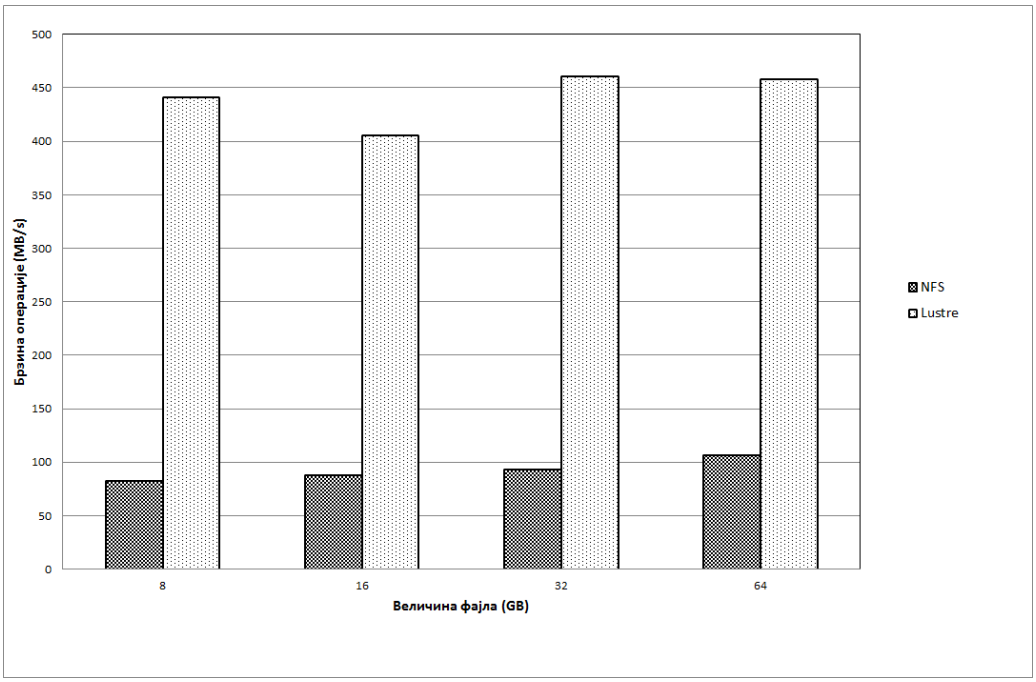
Добијени су следећи резултати:



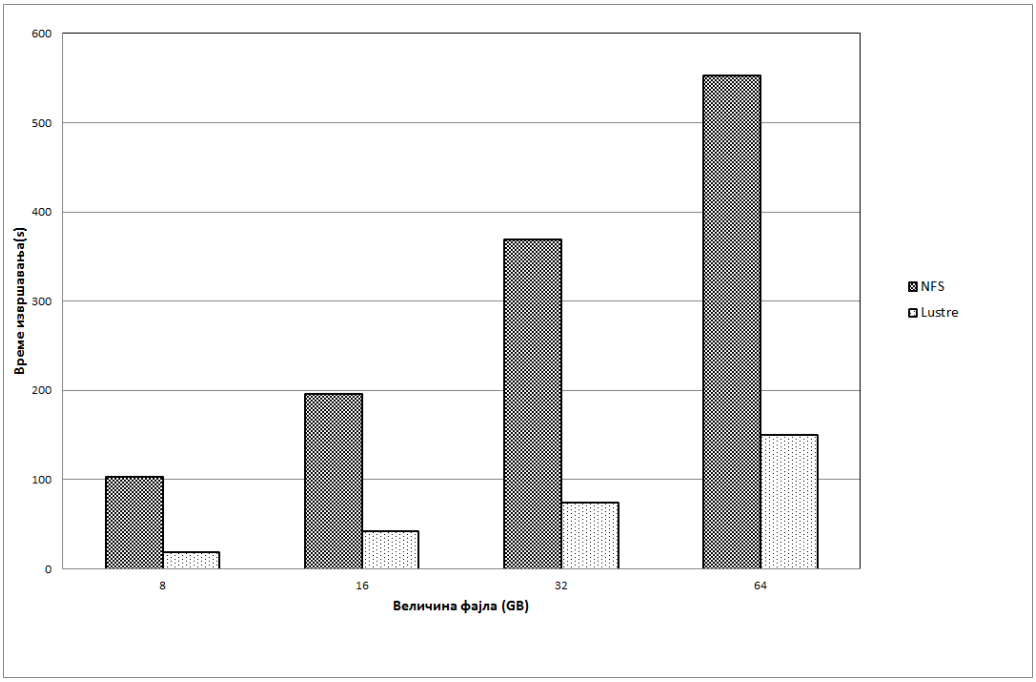
Слика 2.3: Резултати брзине читања



Слика 2.4: Резултати времена извршавања програма читања



Слика 2.5: Резултати брзине писања



Слика 2.6: Резултати времена извршавања програма писања

Анализирајући слике 2.3 и 2.5 на којима је приказана брзина читања односно писања закључује се да је *Lustre* фајл систем вишеструко бржи независно од величине фајла који се чита, односно који се уписује. На сликама 2.4 и 2.6, које приказују време извршавања програма читања, односно писања, уочава се да је разлика у овим фајл системима већа

уколико је величина фајла већа. Што је фајл већи, *Lustre* показује већу супериорност у односу на NFS.

2.4 Game of Life

Game of Life је најпознатији пример ћелијског аутомата који је осмислио британски математичар Џон Конвеј 1970 године. Ова игра је игра без играча, што значи да је њена еволуција одређена првобитним стањем. За тестирање је коришћен *Game of life* MPI-2 програм, коме су дадате функције читања и уписа стања ћелија, које се позивају након сваког корака симулације. Програм је покретан на *Lustre* фајл систему, а затим са истим улазним параметрима на NFS фајл систему. *Game of Life* је погодан јер не захтева додатне корисничке уносе након покретања програма. Матрица помоћу које се прате стања ћелија могуће је декомпоновати независно од броја процеса. Подешавајући улазне параметре, мења се и величина података коју је потребно уписати у фајл.

2.4.1 Правила

Game of Life је представљена бесконачном дводимензионалном мрежом квадратних ћелија од којих је свака у једном од два могућа стања: активном или пасивном. Свака ћелија је у односу са осам суседних ћелија, које су са њом повезане хоризонтално, вертикално или дијагонално. Са сваким кораком у времену јављају се следеће транзиције:

- Било која активна ћелија која има мање од две активне ћелије које су јој суседне постаје пасивна.
- Било која жива ћелија са више од три активне ћелије које су јој суседне постаје пасивна да не би дошло до пренатрпаности.
- Било која ћелија са две или три активне суседне ћелије живи, не мења се и преноси на следећу генерацију.
- Било која пасивна ћелија која има три активне суседне ћелије постаће и сама активна.

Почетни модел представља почетну популацију система. Свака популација је чиста функција претходне. Правило се наставља примењивати узастопно ради креирања наредних генерација.

2.4.2 Порекло

Конвеј је био заинтересован за проблем представљен четрдесетих година двадесетог века од стране реномираног математичара Џон ван Нојмана који је покушавао да пронађе машину која би могла да изради копије себе и успео је када је пронашао математички модел за такву машину са веома компликованим правилима на правоугаоној мрежи. Игра је доживела своје прво јавно појављивање у октобру 1970-е у колумни „Математичке игре“ под називом фантастичне комбинације Џона Конвеја. Са теоретске тачке гледишта је занимљиво, јер има моћ универзалне Тјурингове машине. Све што се може алгоритамски обрачунавати може се израчунати и са Конвејевом *Game of Life*.

Још од њеног објављивања, *Game of Life* је привукла велико интересовање због изненађујућих начина на који се обрасци могу развијати. Занимљиво је за физичаре, биологе, економисте, математичаре, филозофе, научнике и остале да посматрају начин на који се сложени обрасци могу појавити из примене веома једноставних правила. На пример, филозоф и научник Даниел Ц. Денет је користио Конвејеве *Game of Life* интензивно да илуструје могућу еволуцију сложених филозофских конструкција, као што су свест и слободна воља,

од релативно једноставног скупа детерминистичких физичких закона који регулишу наш сопствени универзум. Популарност Конвејеве игре је потпомогнута њеном појавом баш у време појаве нове генерације јефтиних мини рачунара који су пуштени у промет. Игра може да се активира ноћу, током сати када су машине иначе неискоришћене. За многе, *Game of Life* је једноставно програмирање, изазов, забаван начин да губимо циклусе процесора. За неке, међутим, *Game of Life* је више филозофске конотације.

2.5 Опис програма

Програм је написан у C програмском језику користећи MPI функције. На почетку програма се учитавају параметри програма.

Улазни параметри програма су:

- начин генерисања почетне популације
- број процеса
- величина квадратне матрице
- број итерација
- начин уписа - уколико је 0 онда се матрица уписује у 1 фајл. Уколико је 1 онда сваки процес свој део матрице уписује у свој фајл.

Свака ћелија је представљена као поље у матрици. На основу унетих параметара, креира се и генерише почетна популација. У свакој итерацији се на основу објашњених правила рачуна вредност поља у матрици, а затим се на основу начина уписа резултат уписује у фајл или фајлове. Као резултат програма, добија се време које је потребно да се програм изврши.

На кластеру програм се покреће помоћу следеће скрипте:

Листинг 2.2: Скрипта за покретање програма на кластеру

```
#!/bin/sh
#PBS -N lustre_mpi
#PBS -q batch
#PBS -l nodes=8:ppn=8

module load openmpi-pmf-x86_64
chmod 755 mpi_lustre
mpirun ./mpi_lustre random 128 128 1
```

пис и читање из фајла се врши у свакој итерацији. У зависности од начина уписа, потребно је отворити фајл или фајлове за читање и писање. Уколико је `write_type` једнак нули онда сви процеси врше операције са једном фајлом, у супротном сваки процес има посебан фајл из ког чита и у који уписује.

Листинг 2.3: Део кода за отварање фајлова за читање и писање

```
MPI_File thefile;
int nnp, *myold;
MPI_Status status;

  if (write_type == 0)
  {
      MPI_File_open(MPI_COMM_WORLD, filename,
                    MPI_MODE_CREATE | MPI_MODE_RDWR,
                    MPI_INFO_NULL, &thefile);

      MPI_File_set_view(thefile, nnp * id * sizeof(int),
```

```
                                MPI_INT, MPI_INT, "native", MPI_INFO_NULL)
                                    ;
        }
        else
        {
            MPI_File_open(MPI_COMM_SELF, filename,
                          MPI_MODE_RDWR | MPI_MODE_CREATE, MPI_INFO_NULL
                              , &thefile);
        }
```

Листинг 2.4: Функције за читање и писање у фајл

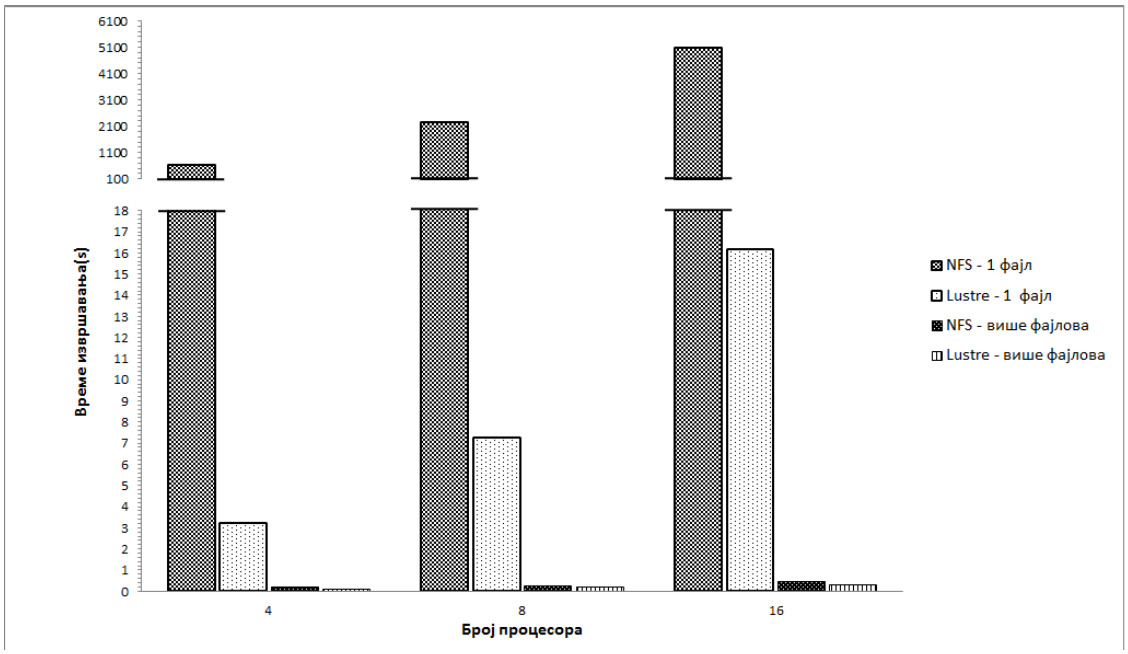
```
    MPI_File_read(thefile, myold, nnp, MPI_INT, &status);

    MPI_File_write(thefile, myold, nnp, MPI_INT, MPI_STATUS_IGNORE);
```

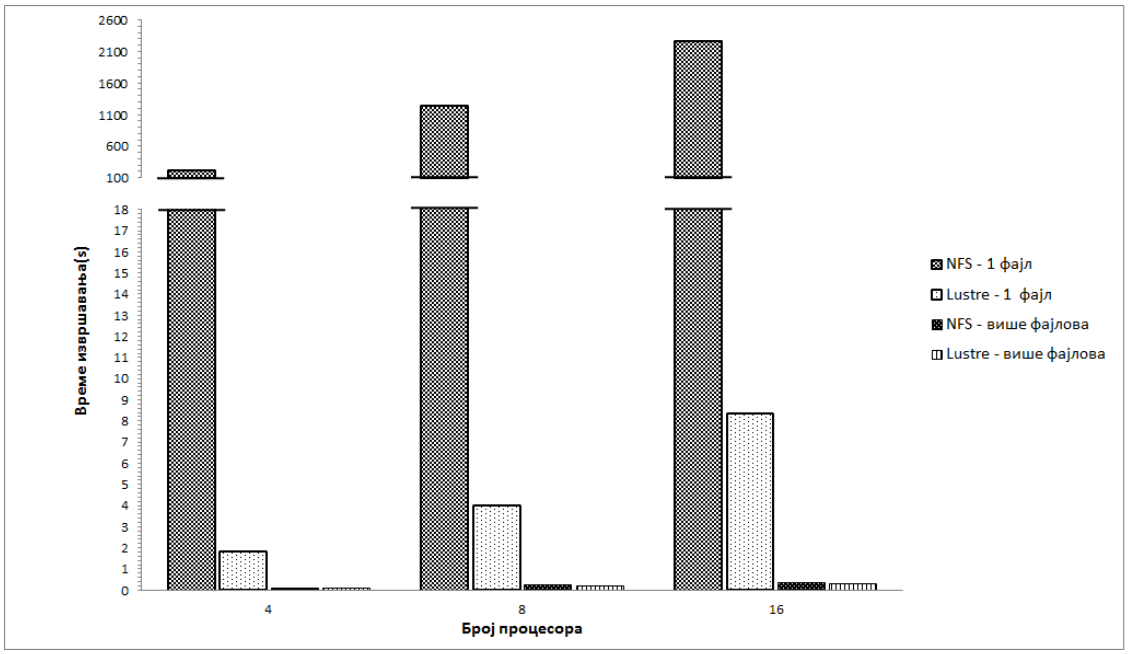
где се садржај низа `myold` чита или уписује у фајл `thefile`.

2.6 Резултати тестирања

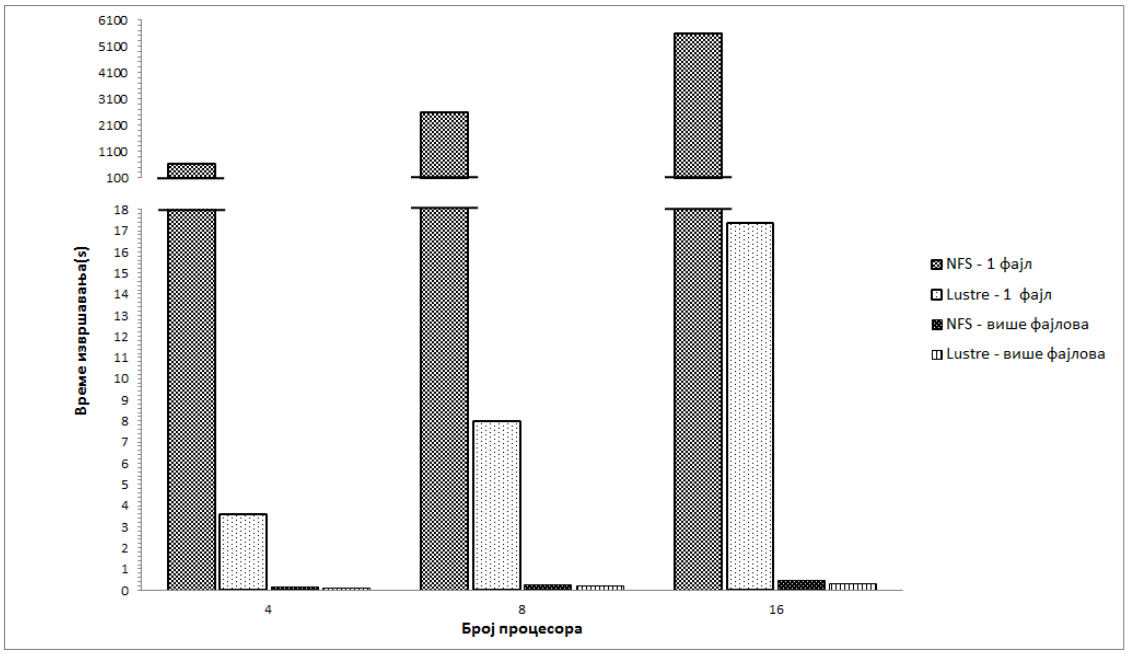
На графиконима је приказано време извршавања у зависности од улазих параметара.



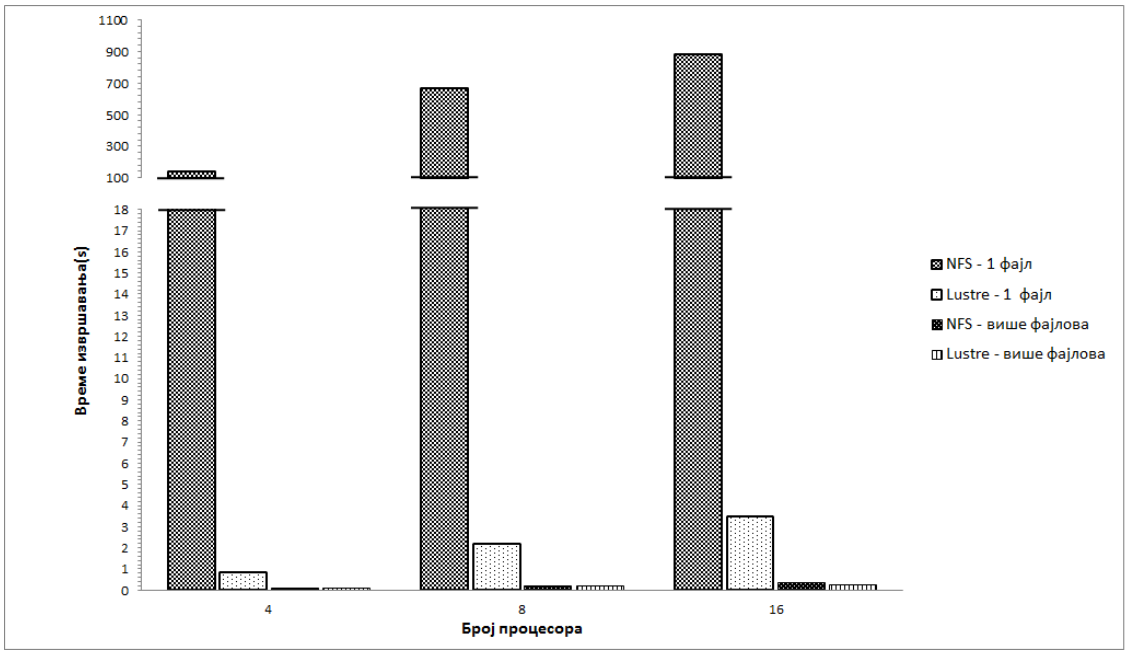
Слика 2.7: Матрица 32x32, 32 итерација



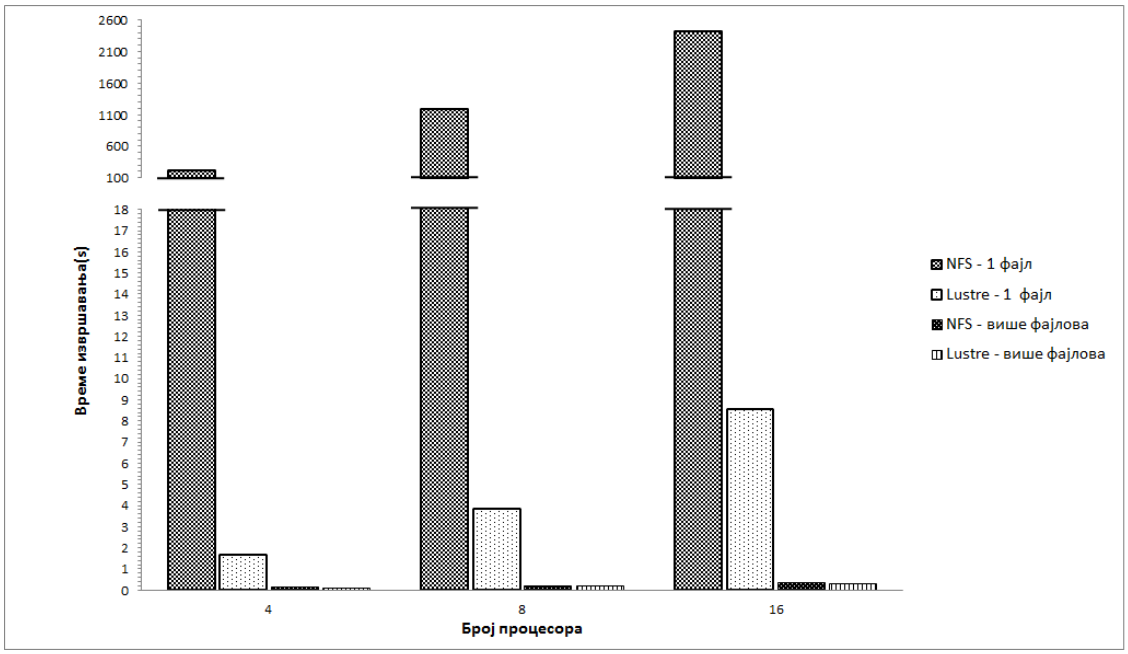
Слика 2.8: Матрица 32x32, 64 итерација



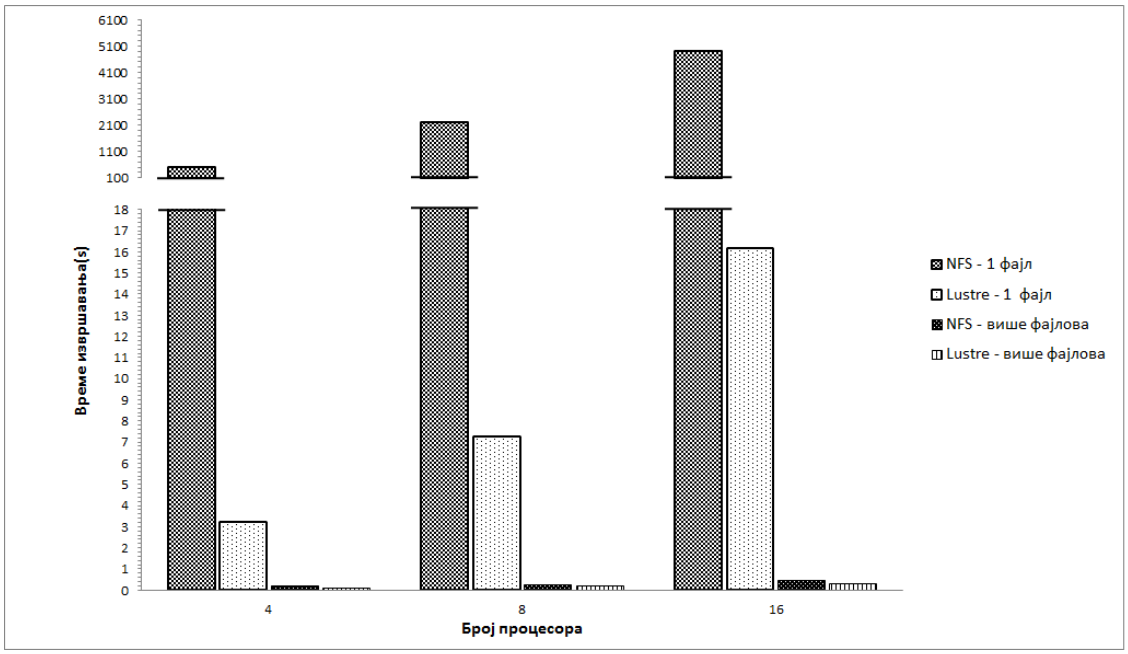
Слика 2.9: Матрица 32x32, 128 итерација



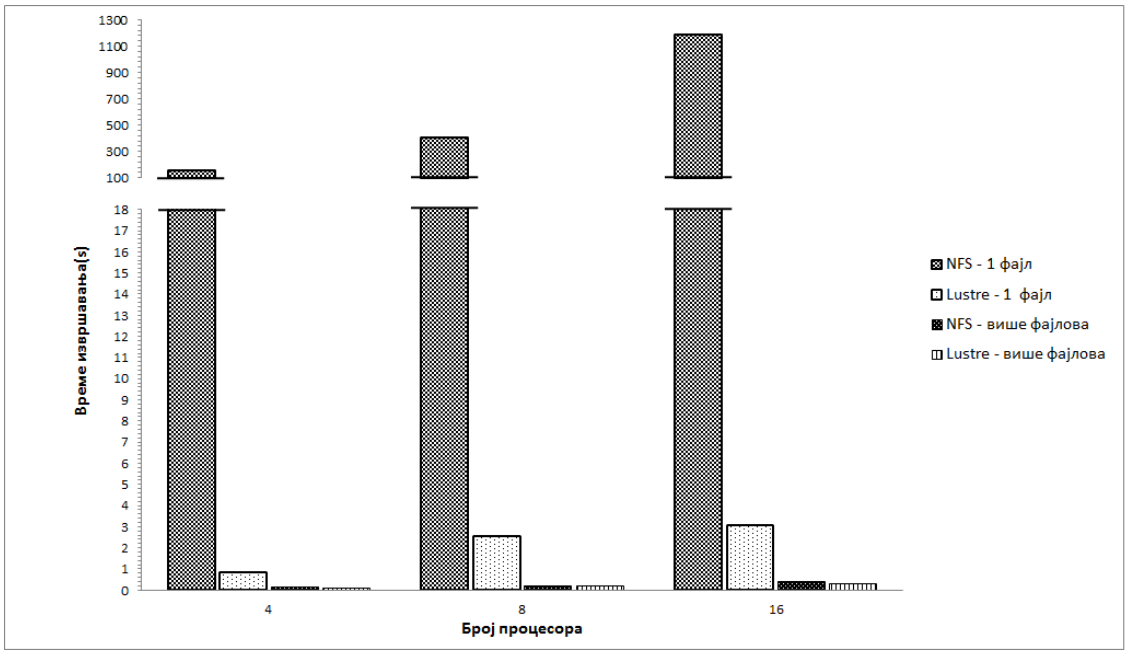
Слика 2.10: Матрица 64x64, 32 итерација



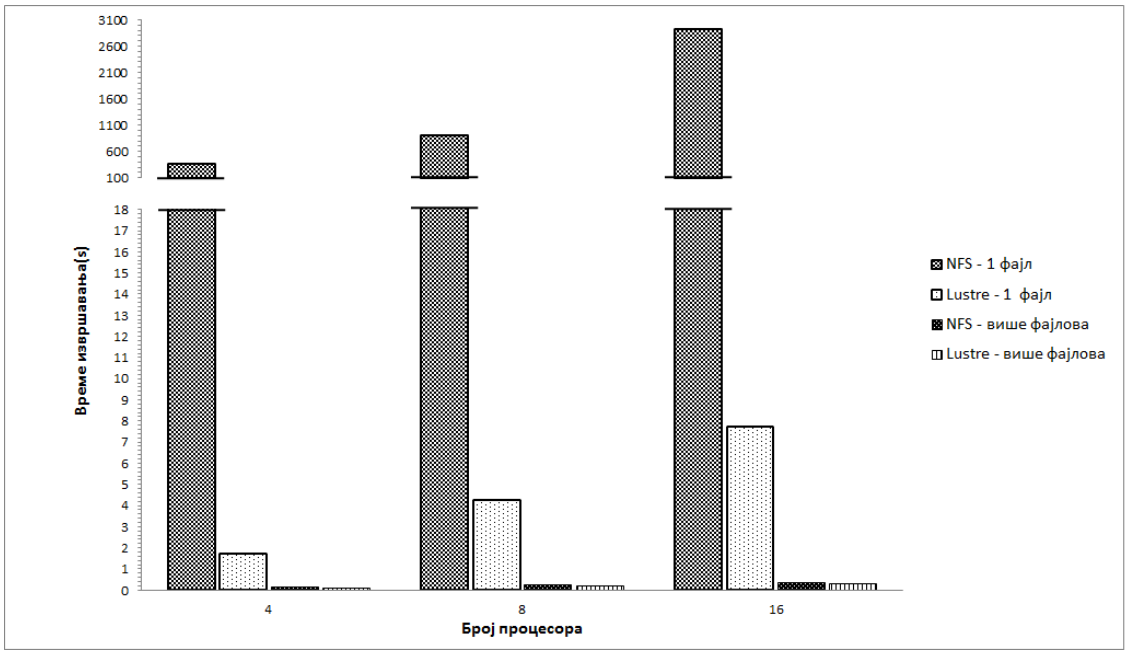
Слика 2.11: Матрица 64x64, 64 итерација



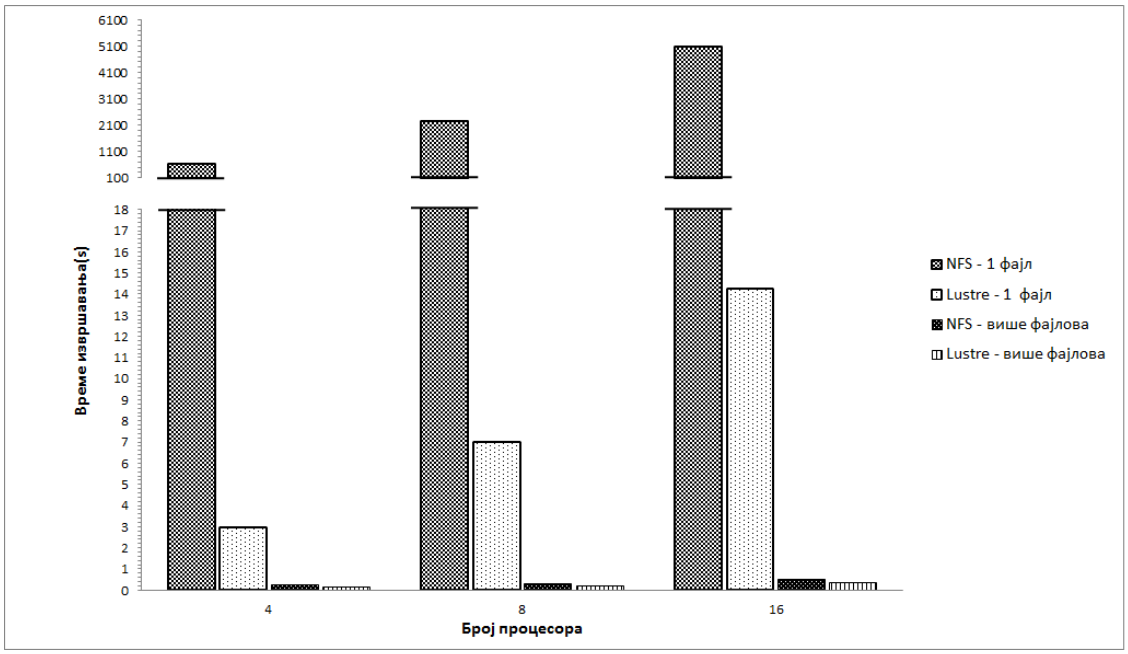
Слика 2.12: Матрица 64x64, 128 итерација



Слика 2.13: Матрица 128x128, 32 итерација



Слика 2.14: Матрица 128x128, 64 итерација



Слика 2.15: Матрица 128x128, 128 итерација

Од слике 4.7 до 4.15 приказано је време извршавања програма *Game of Life* у зависности од комбинације параметара као што су: број процеса, величине матрице, број итерација и начин коришћења улазно/излазних операција. Уочава се да је време извршавања значајно мање код *Lustre* фајл система, посебно када сваки процес има засебан фајл. Када сви процеси користе један фајл, *Lustre* фајл систему је опет потребно мање времена него NFS фајл систему. Повећањем броја процеса ова разлика постаје уочљивија. Што је већа матрица и већи број итерација, то је количина података већа. Читањем и писањем већег броја података, *Lustre* показује већу ефикасност.

Глава 3

Закључак

Тестирањем брзина улазно/излазних операција код *Lustre* фајл система и NFS добијени су резултати који показују да је *Lustre* фајл систем далеко бржи у свим типовима операција. Главни недостатак *Lustre* фајл система је компликована инсталација у односу на његову алтернативу NFS. Међутим, уштеда времена у току вршења тестирања је огромна што се никако не сме занемарити. Та предност је нарочито значајна при извршавању обимних математичких операција које захтевају рад са фајловима. MPI-2 стандард са новим функцијама које омогућавају паралелне улазно/излазне операције заједно са *Lustre* фајл системом чине најбољу комбинацију за паралелне програме. Обзиром да је *Lustre* фајл систем доступан бесплатно, да је његов опоравак лакши у случају отказивања чврстог диска и да подржава паралелне операције са фајловима, треба му се дати апсолутна предност у односу на NFS. Једина алтернатива коју *Lustre* тренутно има на тржишту је *PanFS* компаније *Panasas*, који се нешто лакше конфигурише и администрира, али по цену затворености кода и више цене.

Библиографија

[1] http://stormerider.com/wp-content/uploads/2011/05/820-7390.pdf, март 2014

[2] http://docs.oracle.com/cd/E19527-01/821-2076-10/821-2076-10.pdf, мај 2014

[3] http://linux.cloudibee.com/2009/11/lustre-cluster-filesystem-quick-setup-guide/, октобар 2014

[4] https://www.kernel.org/doc/ols/2003/ols2003-pages-380-386.pdf, октобар 2014

[5] http://imi.pmf.kg.ac.rs/moodle/file.php/127/Materijali_za_vezbe/2013_14_Primeri/13_GOL.c, мај 2015

[6] http://www.conwaylife.com/wiki/Conway's_Game_of_Life, мај 2015

[7] Using MPI-2 Advanced Features of the Message-Passing Interface, William Gropp, Ewing Lusk, Rajeev Thakur, 1999

[8] MPI – the Complete Reference. Vol. 2, The MPI-2 Extensions Scientific and Engineering Computation Series Gropp, William. MIT Press, 1998

[9] MPI-2: Extensions to the Message-Passing Interface - Message Passing Interface Forum, 2009

[10] Recent Advances in Parallel Virtual Machine and Message Passing Interface: 14th European PVM/MPI User's Group Meeting, Paris France, 2007