Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.9
DOI: https://doi.org/10.24867/12BE41Cukovic

ЈЕДАН ПРИМЈЕР АНАЛИЗЕ СИСТЕМА ТОЛЕРАНТНОГ НА ОТКАЗЕ

AN EXAMPLE OF FAULT TOLERANT SYSTEM ANALYSIS

Ивана Ћуковић, *Факултет техничких наука, Нови Сад*

Oblast – ЕЛЕКТРОТЕХНИКА И РАЧУНАРСТВО

Kratak sadržaj – *Овај рад се бави истраживањем поузданости сервера задуженог за прикупљање и пренос података приликом примјене вруће резерве.*.

Ključne reči: *Поузданост сервера, Редуданса, Врућа резерва*

Abstract – *This paper examines reliability of server which is used for collecting and transmission of data on which is applied hot-standby redundancy.*

Keywords: *Reliability of server, redundancy, hot-standby redundancy*

1. УВОД

Сваки софтвер је подложан многим грешкама које могу бити предвиђене или непредвиђене. Свака грешка на софтверу са собом носи одређене посљедице.

Циљ сваке компаније је да произведе рјешење које је толерантно на отказе. С обзиром на то да постоји велики број типова отказа, то није тако једноставно. Испитивањем типова отказа, дошло се до закључка које особине сваки софтвер треба да има како би био толерантан на отказе. Једна од тих особина на коју ће се посебно обратити пажња је поузданост. Поузданост се карактерише као способност континуалног извршавања без прекида. До постизања ове особине се може доћи на различите начине, а један од тих начина је примјена технике редундансе

Редуданса се огледа у дуплирању неког сервера, како би у случају његовог пада неки од тих дупликата могао да преузме његову улогу и да настави са радом. Број сервера који ће се налазити у позадини, као и начин њиховог рада даље зависе од типа редундансе која је примјењује. Као предмет истраживања овог рада посматраће се врућа резерва, као и њене предности и недостаци.

1.1. Грешка и узрок грешке

Веома често се грешка (*енгл. error*) и узрок грешке, односно отказ, (*енгл. fault*) користе у истом контексту, иако у њиховом значењу постоје знатне разлике.

Узрок грешке се односи на дефекте на хардверу или на неку софтверску грешку (*енгл. bug*), док грешка представља манифестацију тог узрока или *bug*-а [1].

Хардверски узроци грешака могу бити класификовани као пролазни (*енгл. transient*), испрекидани (*енгл. intermittent*) и стални (*енгл. permanent*). Поред ове подјеле, хардверски узроци грешака се још могу подијелити на добронудне (*енгл. benign*) и злонамјерне (*енгл. malicious*) [1].

1.2. Типови отказа и толерантност на отказе

Постоје различити типови отказа [2]:

- Отказ у сличају пада (сервер се зауставља, али је до тада радио коректно)
- Отказ у случају пропуста (сервер отказује код одговора на захтјеве клијената)
  - Пропуст при пријему (сервер отказује код пријема порука)
  - Пропуст при слању (сервер отказује код слања порука)
- Отказ због временског ограничења (одговор од сервера касни дуже од специфичног периода)
- Отказ при одговору (одговор сервера је некоректан)
  - Отказ због вриједности (вриједност у одговору сервера је лоша)
  - Отказ због промјене стања (сервер одступа од стандарних активности)
- Произвољан или византијски отказ (сервер може дати произвољан одговор у произвољно вријеме).

Толерантност на отказе представља способност система да настави извршавање без прекида у случају да једна или више његових компоненти откаже [1]. Системи који су толерантни на отказе морају да испуњавају сљедеће:

- доступност,
- поузданост,
- сигурност и
- одрживост.

1.3. Редуданса

Једна од основних особина које купац захтјева за свој поризвод од произвођача је поузданост. Поузданост се може постићи на разне начине, као што су побољшање поузданости компоненти или примјена техника редундансе.

Редуданса представња главни метод за постизање поузданости и сигурности система [3]. Додавањем редундансе се повећава цијена и комплексност система. Опште прихваћена чињеница је да се не може постићи поузданст система отпорног на грешке

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Дарко Чапко, ванр. проф.

без употребе редудансе. Постоје два основна типа редудансе, активна редуданса и резерва (*енгл. standby redundancy)* [4]. На слици 1 можемо видјети графички приказ подјеле редудансе

Слика 1. *Графички приказ подјеле редудансе*

Активну редудансу карактерише то што су све исправне редундантне компоненте у оперативном стању све вријеме, чак иако је у том моменту за нормално функционисање система потребно само њих неколико.
У овом случају нема промјене на исправну комроненту или покретања нове исправне компоненте након пада друге [5].

За разлику од активне редудансе, резерву карактерише оперативно стање компоненти само у случају када постоји недостатак компоненти које су у том стању, а неопходне су за успјешно функционисање система. Резерва се даље може класификовати као хладна (*енгл. cold*), топла (*енгл. warm)* и врућа (*енгл. hot*) [4].

Хладна резерва подразумијева коришћење редундантне компоненте на начин да она буде заштићена од стреса који носи оперативно стање. Без изложеност овом стресу, вјероватноћа испада те компоненте је веома ниска, претпоставља се да је приближна нули, све до момента када мора да постане оперативна и замијени компоненту на којој се десио испад [6].

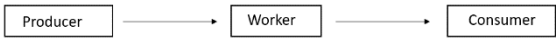
Топлу резерву карактерише степен вјероватноће испада редундантне компоненте који је између степена вјероватноће хладне и степена вјероватноће вруће редудансе приправности, која ће бити детаљно објашњена у наредном поглављу. Хладна, као и врућа резерва, представљају један од облика топле резерве [7].

Када редундантна компонента ради у паралели са примарном компонентом, обављајући идентичан посао и посједовајући исте податке, спремна да у сваком моменту преузме примарну улогу, говоримо о врућој резерви. Овај облик редудансе обезвјеђује брз опоравак након испада примарне компоненте, али и додатне операције и трошкове будући да троши енергију и материјале.

С обзиром да се налази под потпуним стресом који носи оперативно стање, на овој компоненти може да се деси испад и прије него што она замјени примарну компоненту, односно степен вјероватноће испада редуданте компоненте је једнак степену вјероватноће испада примарне компоненте [7].

2. ФОРМУЛАЦИЈА ПРОБЛЕМА И ПРЕДЛОЖЕНО РЈЕШЕЊЕ

Сваки прекид у раду система може да изазове велике посљедице које је тешко исправити. Посматране су посљедице прекида рада система за прикупљање и прослеђивање података у електроенергетском систему.

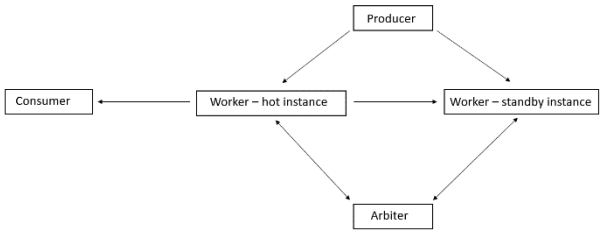


Слика 2 – *Архитектура основног система*

На слици 2 је приказана архитектура упрошћеног система за прикупљање и прослеђивање података у електроенергетском систему. Систем се састоји од три основне компоненте:

- *Producer*, компонента која шаље податке
- *Worker*, средишња компонента која прикупља и даље просљеђује податке
- *Consumer*, крајња компонента којој се испоручују прикупљени подаци на даљу обраду.

Проблем који се јавља приликом овог рјешења је у случају испада на *Worker*-у. Када компонента престане да ради, подаци са паметних мјерила не долазе до *Consumer*-а до његовог поновног подизања, а подаци који су стигли прије пада, а нису прослијеђени се губе. Количине података које се изгубе приликом оваквог испада компоненте могу да изазову велике проблеме, услијед неажурних података у електроенергетском систему.



Слика 3 – *Архитектура предложеног софтверског рјешења*

На слици 3 је приказана архитектура предложеног софтверског рјешења како би се избјегли проблеми губљења података.

Компонента *Worker* добија своју копију, односно креира се још једна њена инстанца. Двије инстанце се разликују по томе што се при покретању једна поставља за примарну (енгл. *hot instance*), а друга за редундантну инстанцу (енгл. *standby instance*).

Producer у овом рјешењу комуницира и са примарном и са редундантном инстанцом истовремено, шаљући им исте податке. Разлог је управо то што врућу резерву карактерише да обје њене инстанце имају исте податке.

Иако имају исте података, остатак посла не извршавају на исти начин. Обје инстанце прихватају исте податке, чувају их, али када дође до дијела да се ти подаци прослеђују даље, то ради само примарна компонента. Слањем података само са примарне компоненте се избјегава дуплирање података на пријемној страни. Поред тога што примарна инстанца шаље податке ка компоненти *Consumer*, она обавјештава редундантну компоненту о идентификационом броју података који је последњи прослиједила, како би

редундантна инстанца, у случају да постане примарна, знала од ког дијела треба да настави слање. Подразумјева се да сваки податак који прође кроз систем има јединствен идентификациони број.

Arbiter води рачуна која инстанца ће бити примарна,а која редундантна. *Arbiter* покреће обје инстанце компоненте *Worker*. Инстанца која се прва покрене добија примарну улогу, а друга редундантну. Након покретања примарне и редундантне инстанце, њен задатак је да на одређен временски период „прозива“ обје и да оне узвраћају на тај позив како би знала да су још увијек активне. У овом конкретном случају дефинисани период је једна секунда.

3. ТЕСТИРАЊЕ

Тестирано је предложено рјешења са 5 инстанци компоненте *Producer* које шаљу фиксан број података у једној секунди. Тестови су урађени за 10.000, 15.000 и 20.000 података који стижу са компоненте *Producer* у једној секунди, при чему се тестови извршавају док број послатих података не достигне 1.000.000. Подаци који се шаљу у секунди од стране компоненте *Producer* су груписани у пакете, не шаљу се појединачно.

Пад примарне инстанце компоненте *Worker* је симулиран гашењем те инстанце у произвољном временском тренутку. Будући да се ради о веома малим јединицама времена у којима се све извршава, немогуће је одредити тачан временски моменат у коме желимо да компонента престане са својом функционалношћу.

Тестирањем су обухваћени случајеви до којих може да се дође примјеном овог рјешења.

3.1. „Идеалан“ случај примјене предложеног рјешења

„Идеалним“ случајем можемо сматрати случај када након пада примарне инстанце, редундантна одмах преузима њену улогу и наставља са радом. Под овим се подразумијева да је примарна инстанца оборена у моменту након што је прослиједила податке примљене у претходној секунди, као и да је након слања тих података обавјестила редундантну инстанцу о идентификационом броју последњег послатог података. Након преузимања примарне улоге, инстанца зна тачно од ког података треба да настави слање.

Можемо посматрати два различита периода рада овог предложеног рјешења. Један период је прије пада примарне компоненте, а други након њеног пада и преузимања њене улоге од стране редундантне компоненте.

Први посматрани случај је слање 10.000 података са 5 инстанци. Просјечно вријеме које је потребно да подаци стигну од компоненте *Producer* до компоненте *Consumer*, у случају када је редундантна компонента ниједном није преузела улогу примарне, је 981,6 милисекунди са стандардном девијацијом од 309 милисекунди. Када посматрамо просјечно вријеме преноса података након што је редундантна компонента преузела улогу примарне, можемо закључити да нема велике разлике. Просјечно вријеме преноса у том случају је 876 милисекунди са стандардном девијацијом од 319 милисекунди. За пренос свих података, просјечно вријеме трајања преноса података износи 906,7 милисекунди са стандардном девијацијом од 319,6 милисекунди.

Како би показали да овај случај није примјењив само када је број података мањи, тестиран је случај када се са истог броја инстанци компоненте *Producer* шаље дупло већи број података. Свака инстанца у периоду од једне секунде шаље 20.000 података.

Како би показали сличност са претходним примјером, посматра се вријеме које је потребно за пренос података прије пада примарне инстанце, и након њеног пада и преузимања рада од стране редундантне инстанце. Прије пада примарне инстанце просјечно вријеме слања података је 1,99 секунди са стандардном девијацијом од 448 милисекунде, а након пада 1,49 секудни са стандардном девијацијом од 407 милисекунди. Можемо примјетити да временска разлика између првог и другог периода није значајна и да је веома слична случају када се шаље 10.000 података. Просјечно вријеме преноса података је 1,84 секунди са стандардном девијацијом од 494 милисекунде.

Када посматрамо просјечно вријеме преноса података у случају са 10.000 података и у случају са 20.000 података видимо одређену разлику. Међутим, та разлика је управо због количине података која се преноси.

3.2. Веће кашњење проузроковано нагомилавањем података

Да би боље разумјели овај случај потребно је добро познавати рад редундантне компоненте. Када се покрене инстанца компоненте *Worker* она прима податке, складишти их и одбројава временски период од једне секунде након чега се иначе врши слање прикупљених података. Међутим, редундантна инстанца компоненте након истека периода од једне секунде не ради ништа. Задржава податке које има док год јој примарна компонента не пошаље идентификациони број последњег послатог података, а затим брише све податке који су стигли прије њега, као и њега самог.

У овом конкретном случају примарна компонента је „оборена“ тако да у неком временском периоду ниједна компонента није имала улогу примарне. Управо због тога је редундантна компонента, када је постала примарна, имала већи број скупљених података које је требало прослиједити. Тај већи број података, који се разликује од просјечног броја података послатих у претходном периоду, доводи до кашњења које затим утиче на све наредне пакете података. Подаци се и даље шаљу сваке секунде, али компонента *Consumer* их не преузима тако брзо.

За разлику од случаја описаног у претходном примјеру, у овом случају ћемо видјети како број података који се шаље значајно утиче на перформансе предложеног рјешења.

Посматрањем резултата добијених тестирањем када се шаље 10.000 података закључује се да након промјене улоге примарне и редудантне компоненте, број података је већи од просјечног броја који се шаље. У даљем наставку рада се тај број пакета враћа на просјечан. То је управо посљедица што у периоду од једне секунде ниједна од ове двије инстанце компоненте *Worker* није имала улогу примарне и долази до нагомилавања података.

Будући да се ради о не тако великој количини података просјечно вријеме слања података, које износи 1,12 секунди са стандардном девијацијом од 413 милисекунди, је и даље приближно просјечном времену слања у нормалним условима. Ако посебно посматрамо период прије пада примарне компоненте, гдје је просјечно вријеме 965,7 милисекунди са стандардном девијацијом од 314 милисекунди, и период након њеног пада, гдје је просјечно вријеме 1,21 секунда са стандардном девијацијом од 433 милисекунди, ту примјетимо одређену разлику. Та разлика је изазвана временом потребним да се пренесе први пакет података након што је редудантна компонента преузела улогу примарне.

Када се посматра случај слања 20.000 података са 5 инстанци, може да се примјети значајнија временска разлика. Када посматрамо само период прије пада примарне компоненте, видимо да је просјечо вријеме преноса података 2,26 секунди са стандардном девијацијом од 612 милисекунди, док је након њеног пада то вријеме много веће, 5,18 секунди са стандардном девијацијом од 743 милисекунде. Као и у претходном примјеру, ово кашњење након пада је изазвано обимом података у првом пакету. Тај пакет је до 3 пута већи од свих послатих прије и каког њега.

3.3. Дуплирање података

За овај случај тестирања је извршено слање 15.000 података са 5 инстанци компоненте *Producer*.

У овом случају примарна инстанца компоненте *Worker* је „оборена“ у моменту када је податке које је скупљала секунду прослиједила даље, али није стигла да јави редудантној компонентни идентификациони број последњег послатог податка. Након преузимања примарне улоге, редудантна компонента, након истека временског периода од једне секунде, даље прослеђује све податке који се налазе код ње. Управо због овога долази до дуплирања података на пријемној страни, а из примјера можемо видјети да је послато 60.000 дуплираних података.

Дуплирање података у овом случају са собом повлачи проблем из примјера описаних у поглављу 3.2. Када посматрамо период прије пада примарне компоненте, просјечно вријеме преноса података је 1.46 секунди са стандардном девијацијом од 375 милисекунди, а након њеног пада и преузимања њене улоге од стране редудантне компоненте, просјечно вријеме је 5,09 секунди са стандардном девијацијом од 701 милисекунде. У овом примјеру, просјечно вријеме за пренос података у току цјелокупног рада је 3,63 секунде са стандардном девијацијом од 1,87 секунди.

4. ЗАКЉУЧАК

Примјена вруће резерве умногоме повећава поузданост система. Обезбјеђује његов континуалан рад у великом броју случајева. Наравно, за неке случајеве се та поузданост не може осигурати. Највећи недостатак редудансе је могућност отказа додатних компоненти које се користе за активирање компоненте из стања резерве када је то потребно.

Основна предност се огледа управо у томе што се падом примарне инстанце у било ком моменту обезбјеђује да сви подаци стигну на одредиште. Свака предност са собом носи и одређене недостатке. У овом случају ти недостаци су представљени дуплирањем података, као и додатним кашњењем приликом испоруке. Међутим, када се предност упореди са дуплирањем и кашњењем података, јасно је да та два недостатка могу да се занемаре. Посљедице које они изазивају су далеко мање од посљедица изазваних губитком података у системима са критичном мисијом.

5. ЛИТЕРАТУРА

[1] Israel Koren and C. Mani Krishna, „*Faule-Tolerant Systems“*, Morgan Kaufmann, 2020.

[2] Andrew S. Tanenbaum, „*Distributed Systems Principles and Paradigmas“*, Prentice-Hall, 2007.

[3] Z. Wang, Y. Chen, W. Men and R. Kang, “Failure behavior analysis of hot standby system based on BDD method”, *Safety and Reliability–Safe Societies in a Changing World* , pp. 2441-2448, CRC Press, 2019.

[4] Suprasad V. Amari and Glenn Dill, “A new method for reliability analysis of standby system”, IEEE *Annual Reliability and Maintainability Symposium,* pp. 417-422, Relex Software Corporation, 2009.

[5] W. Kuo, V.R.Prasad, F.A.Tillman, C.L.Hwang, *“Optimal reliability Design: Fundamentals and Applications”*, Cambridge university press, 2001.

[6] David W. Colt, Cold-standby redundancy optimization for nonrepairable systems, *Iie Transactions*, Vol. 33, No.6, pp. 471-478, 2001.

[7] Suprasad V. Amari, Hoang Pham and Ravindra B. Misra, “Reliability Characteristics of k-out-of-n Warm Standby Systems”, *IEEE Transactions on Reliability*, Vol.61, No.4, pp. 1007-1018., 2012.

Кратка биографија:

Ивана Ћуковић рођена је у Корану 1995. године. Дипломирала је на Факултету техничких наука из области Електротехнике и рачунарства – Примјењено софтверско инжењерство 2018. године.
контакт: icukovic14@gmail.com