UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Andrej Kristanc

ZAGOTAVLJANJE VISOKE RAZPOLOŽLJIVOSTI VIRTUALNE KNJIŽNICE PRI IMPLEMENTACIJI SISTEMA IBM TS7650G V GRUČI WSFC

DIPLOMSKO DELO

VISOKOŠOLSKI STROKOVNI ŠTUDIJSKI PROGRAM PRVE STOPNJE
RAČUNALNIŠTVO IN INFORMATIKA

Mentor: doc. dr. Mira Trebar

Ljubljana, 2011

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00077/2011
Datum: 04.03.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **ANDREJ KRISTANC**

Naslov: **ZAGOTAVLJANJE VISOKE RAZPOLOŽLJIVOSTI VIRTUALNE KNJIŽNICE PRI IMPLEMENTACIJI SISTEMA IBM TS7650G V GRUČI WSFC**

ENSURING HIGH-AVAILABILITY OF VIRTUAL TAPE LIBRARIES BY IMPLEMENTING IBM TS7650G WITH WSFC CLUSTER

Vrsta naloge: Diplomsko delo visokošolskega strokovnega študija prve stopnje

Tematika naloge:

Kandidat naj v diplomskem delu analizira delovanje gruče, ki omogoča replikacijo podatkov na dveh vozliščih TS7650G. Preuči naj njene možnosti uporabe na dveh fizično oddaljenih lokacijah z namenom zagotavljanja visoke razpoložljivosti sistema. Vozlišči TS7650G naj implementira v WSFC (Windows Server Failover Clustering) z uporabo vira Generic Script Resource, pri čemer naj napiše skripte in uporabi različna orodja, ki bodo z vozliščema TS7650G upravljale tako, kot to zahtevajo standardne funkcije vira Generic Script Resource. Za implementirano izvedbo naj bo podan opis delovanja in testiranja ter morebitne izboljšave.

Mentor:

doc. dr. Mira Trebar

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

Dekan:

prof. dr. Nikolaj Zimic

Rezultati diplomskega dela so intelektualna lastnina Fakultete za računalništvo in informatiko Univerze v Ljubljani. Za objavljanje ali izkoriščanje rezultatov diplomskega dela je potrebno pisno soglasje Fakultete za računalništvo in informatiko ter mentorja.

To delo je ponujeno pod licenco *Creative Commons Priznanje avtorstva-Deljenje pod enakimi pogoji 2.5 Slovenija* (ali novejso razlicico). To pomeni, da se tako besedilo, slike, grafi in druge sestavine dela kot tudi rezultati diplomskega dela lahko prosto distribuirajo, reproducirajo, uporabljajo, priobcujejo javnosti in predelujejo, pod pogojem, da se jasno in vidno navede avtorja in naslov tega dela in da se v primeru spremembe, preoblikovanja ali uporabe tega dela v svojem delu, lahko distribuira predelava le pod licenco, ki je enaka tej. Podrobnosti licence so dostopne na spletni strani creativecommons.si ali na Institutu za intelektualno lastnino, Streliska 1, 1000 Ljubljana.

Izvorna koda diplomskega dela, njeni rezultati in v ta namen razvita programska oprema je ponujena pod licenco GNU General Public License, razlicica 3 (ali novejsa). To pomeni, da se lahko prosto distribuira in/ali predeluje pod njenimi pogoji. Podrobnosti licence so dostopne na spletni strani http://www.gnu.org/licenses/.

I Z J A V A O A V T O R S T V U

diplomskega dela

Spodaj podpisani/-a Andrej Kristanc,

z vpisno številko 63050193,

sem avtor/-ica diplomskega dela z naslovom:

Zagotavljanje visoke razpoložljivosti virtualne knjižnice pri implementaciji sistema IBM TS7650G v gruči WSFC

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal/-a samostojno pod mentorstvom doc. dr. Mire Trebar.

- So elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela.

- Soglašam z javno objavo elektronske oblike diplomskega dela v zbirki »Dela FRI«.

V Ljubljani, dne 15.6.2011 Podpis avtorja/-ice:

ZAHVALA

Zahvaljujem se mentorici, doc. dr. Miri Trebar, za nasvete, pomoč in usmerjanje pri pisanju diplomske naloge in Ivu Gomilšku iz podjetja IBM za strokovno pomoč. Zahvaljujem se tudi družini za podporo v času študija.

Kazalo

KRATICE

ARP	Address Resolution Protocol
FC	Fibre Channel
IBM	International Business Machines
MSCS	Microsoft Cluster Server
MAC	Media Access Control
PDU	Power Distribution Unit
SAN	Storage Area Network
SVC	San Volume Controller
UPS	Uninterruptible Power Supply
VTL	Virtual Tape Library
WSFC	Windows Server Failover Clustering
WWPN	World Wide Port Number

POVZETEK

V diplomski nalogi je predstavljena povezava dveh vozlišč TS7650G tako, da dosežemo visoko razpoložljivost (ang. high availability) virtualne tračne knjižnice. To pomeni, da eno od obeh vozlišč TS7650G deluje, drugo pa je v pripravljenosti. Ko delujoče vozlišče TS7650G zaradi kakršnegakoli razloga preneha delovati, se mora v relativno kratkem času avtomatično aktivirati drugo. Z implementacijo v gruči WSFC (Windows Server Failover Clustering) smo dosegli visoko razpoložljivost virtualne tračne knjižnice preko generičnega skriptnega vira (ang. Generic Script Resource). Za upravljanje z obema vozliščema TS7650G smo napisali skripte in na ta način povezali gručo WSFC in obe vozlišči TS7650G. Povezava med gručo WSFC in vozliščema TS7650G deluje tako, da gruča WSFC kliče standardne funkcije, ki so definirane za generični skriptni vir in so zapisane v skriptnem jeziku Visual Basic Script. Te standardne funkcije pa potem kličejo zunanje skripte napisane v skriptnem jeziku Perl, ki preko SSH protokola, upravljajo neposredno z vozliščema TS7650G.

Ključne besede: visoka razpoložljivost, gruča, VTL, TS7650G, MSCS, WSFC

ABSTRACT

Connection of two TS7650G nodes in order to achieve high availability of virtual tape library is presented in the following thesis. This means that one of the two TS7650G nodes is active and the other one is passive. When active TS7650G node for any reason cease to function, the other node must be automatically activated in a relatively short period of time. With the implementation of both nodes into the WSFC (Windows Server Fail-over Clustering) cluster using generic script resource, we have achieved high availability of virtual tape library. By creating special scripts we have linked WSFC cluster with both TS7650G nodes. Relationship between TS7650G nodes and WSFC cluster is working so that WSFC cluster is calling standard functions that are defined for generic script resource and are written in a scripting language Visual Basic Script. These standard functions then call external scripts, written in Perl scripting language, which manage both TS7650G nodes via SSH protocol.

Keywords: high availability, cluster, VTL, TS7650G, MSCS, WSFC

1 UVOD

Vedno večja konkurenca na trgu zahteva od manjših, ozko specializiranih podjetij, pa vse do večjih, mednarodnih podjetij, vedno višji nivo nudenja storitev. Motnje nudenja ali pa pomanjkanje razpoložljivosti storitev za podjetje posledično pomeni izgubo prihodkov, kredibilnosti in priložnosti za pridobivanje novih uporabnikov. Zaradi velike odvisnosti od računalniških sistemov, je nivo nudenja storitev odvisen predvsem od zanesljivosti delovanja le teh, kar pomeni, da bi morali delovati neprekinjeno in brez napak. Kot vemo to v praksi ni izvedljivo, saj do napak in izpadov prihaja, zato je potrebno zanesljivost doseči na drug način. Kritične aplikacije, kot so podatkovne baze, elektronska pošta, spletni strežnik in nekatere druge, se morajo izvajati na sistemu, ki dosega visoko razpoložljivost, saj visoka razpoložljivost sistemov izniči ali pa vsaj ublaži posledice izpadov posameznih komponent ali pa celotnega sistema.

V splošnem je visoka razpoložljivost definirana kot implementacija dizajna sistema, ki zagotavlja visok nivo kontinuitete delovanja v nekem danem obdobju, pri čemer je kontinuiteta delovanja definirana kot nivo storitve, ki jo morajo aplikacije, servisi in sistemi kot celote, zagotavljati končnim uporabnikom, torej nam, našim zaposlenim ali pa strankam. Razpoložljivost je subjektivna, kar pomeni, da je vezana na individualne zahteve v dani situaciji. Visoka razpoložljivost je torej vezana na poslovne cilje in tehnične zahteve, pri čemer je končni cilj čim bolj zmanjšati čas, ko aplikacije, servisi in sistemi kot celote niso razpoložljivi in ne morejo opravljati svoje funkcije.

Doseganje visoke razpoložljivosti se začne pri dizajniranju strojne opreme, na kateri aplikacije in servisi tečejo. To pomeni, da mora biti vsak računalniški sistem izdelan iz čim bolj kvalitetnih komponent in imeti podvojene komponente, ki prevzamejo funkcijo primarne komponente v primeru izpada. V praksi se podvaja komponente, ki so bolj dovzetne za okvare in jih je relativno lahko vključiti v dizajn, to so napajalniki, trdi diski, dinamični spomin, razširitvene kartice, itd. Nekatere druge komponente, kot sta procesor in krmilnik pomnilnika pa je zelo težko podvajati, zato izpade teh komponent rešujemo na višjih nivojih in sicer na nivoju aplikacije in operacijskega sistema. Doseganje visoke razpoložljivosti na nivoju aplikacije in operacijskega sistema je ponavadi izvedeno z replikacijo podatkov med vozlišči in pa z uporabo gruče z uravnoteženim nalaganjem bremena (ang. network load balancing

cluster) ali pa gruče za visoko razpoložljivost (ang. high availability cluster).

TS7650G gruča z replikacijo je sicer zelo dobro zamišljena rešitev, ki pa ima na žalost eno veliko pomanjkljivost. Vozlišči TS7650G v tej gruči morata biti fizično relativno blizu saj sta priklopljeni na isti mrežni PDU element. To predstavlja problem v primeru, da bi radi postavili tak sistem tako, da je eno vozlišče na primarni lokaciji, drugo pa na sekundarni lokaciji, pri čemer sta lokaciji med seboj oddaljeni toliko, da priklop na isti mrežni PDU ni mogoč. Do takega primera pride v praksi precej pogosto, zato je smiselno najti rešitev za ta problem [1].

Rešitev za ta problem je več, mi pa smo ga rešili tako, da smo dve samostojni vozlišči TS7650G implementirali v gruči WSFC in s tem dosegli visoko razpoložljivost virtualne tračne knjižnice. Da smo uspešno implementirali dve samostojni vozlišči TS7650G v WSFC gručo, smo ju morali najprej pripraviti do te mere, da sta bili navzven identični. Ti dve vozlišči TS7650G pa smo potem z uporabo različnih orodij in nekaj skriptami, ki smo jih napisali, povezali v gruči WSFC preko generičnega skriptnega vira.

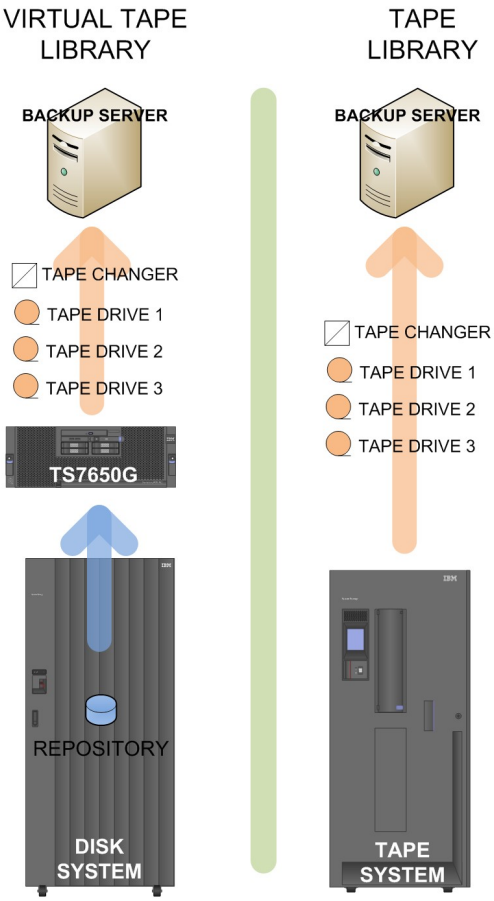
2 TEHNOLOGIJE IN ORODJA

Za delovanje sistema implementacije vozlišč TS7650G v gruči WSFC, smo uporabili tehnologije, ki si jih bomo na kratko ogledali v tem poglavju: (i) TS7650G je sistem, ki smo ga uporabili za zagotavljanje virtualne tračne knjižnice, (ii) FC SAN (Fibre Channel Storage Area Network) skrbi za to, da sta obe vozlišči TS7650G povezani na virtualne diske, na katerih imata repozitorij in na strežnike, ki jim dajeta virtualno tračno knjižnico v uporabo, (iii) SVC (ang. San Volume Controller) je element v FC SAN omrežju, ki virtualizira diskovne sisteme. V našem primeru smo jo uporabili za zagotavljanje virtualnih diskov, na katerih imata vozlišči TS7650G repozitorij, (iv) WSFC v našem primeru skrbi za doseganje visoke razpoložljivosti virtualne tračne knjižnice. Za povezovanje WSFCja z obema vozliščema TS7650G, smo uporabili skriptni jezik Perl in sicer disctribucijo ActivePerl. Zadnja v nizu tehnologij je skupek orodij PsTools, ki smo ga uporabili za izvajanje ukazov v kontekstu lokalnega sistemskega uporabnika.

2.1 TS7650G

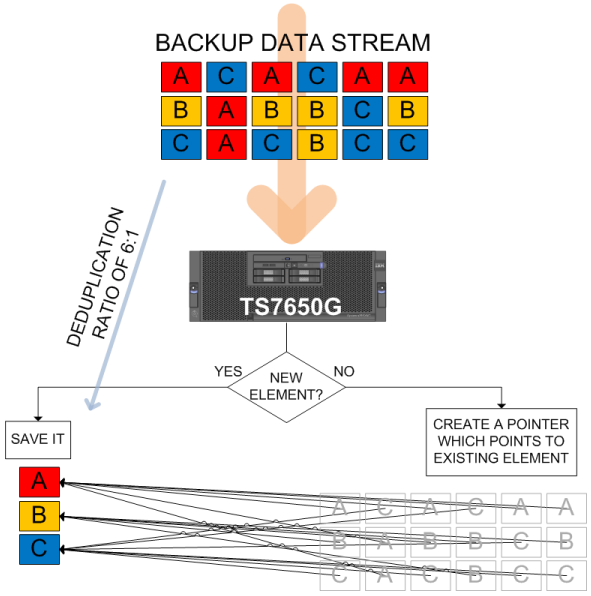
Varnostno kopiranje oziroma arhiviranje podatkov je v računalništvu prisotno praktično že od vsega začetka njegove uporabe. Na začetku se je izvajalo na magnetne trakove, in se v večini primerov še danes, saj so bili magnetni trakovi zaradi svojih karakteristik dolgo časa najboljši medij. Z naraščanjem količine podatkov in predvsem precejšnjega podvajanja podatkov, ali pa vsaj dela podatkov, pa se zadnje čase vse bolj uveljavljajo alternativni pristopi, kot je virtualizacija (ang. virtualization) in deduplikacija (ang. deduplication).

Virtualizacija magnetnih trakov dejansko pomeni, da se naprave, ki upravljajo z virtualnim magnetnim trakom v virtualnem tračnem pogonu dejansko ne zavedajo, da se podatki v ozadju ne shranjujejo na magnetne trakove, pač pa na diskovna polja [1]. Z uporabo virtualizacije pridobimo na zmogljivosti, saj so diskovna polja hitrejša od magnetnih trakov, še posebej če so tudi ta virtualizirana. Primerjava navadne tračne knjižnice z virtualno tračno knjižnico je prikazana na sliki 1.



Slika 1. Virtualizacija.

Deduplikacija je posebna tehnika kompresije, ki zmanjša oziroma izniči prednost, ki jo imajo magnetni trakovi pred diskovnimi polji, to je cenovno ugodnost. Bistvo deduplikacije je, da se podatki ne podvajajo, kar dosežemo s tem, da podatke razdelimo na manjše enote, potem pa shranimo samo enote, ki še niso shranjene [1]. Takoj ko pridemo do enote, ki je že shranjena, naredimo samo kazalec na to enoto, kar pa zasede veliko manj prostora, kot pa če bi morali enak podatek še enkrat shraniti. Brisanje podatkov je izvedeno tako, da se izbriše kazalec na enoto, ki jo brišemo, v primeru, da pa na to enoto ne kaže noben kazalec več, pa se izbriše tudi enota. Faktor deduplikacije je kompresijsko razmerje med količino podatkov pred in po deduplikaciji in je odvisen od vrste podatkov ki jih shranjujemo. V praksi lahko doseže tudi več deset-kratno kompresijsko razmerje. Delovanje deduplikacije je prikazano na sliki 2.



Slika 2. Deduplikacija.

TS7650G je strežnik IBM System x3850 M2, na katerem je nameščen operacijski sistem Red Hat Linux Enterprise Edition verzije 5.4 in programska oprema IBM System Storage ProtecTIER Enterprise Edition, ki virtualizira tračno knjižnico in uporablja IBM HyperFactor tehnologijo za deduplikacijo. Strežnik ima štiri šest-jedrne procesorje Intel Xeon X7460 in 32GB dinamičnega spomina. Za priklop na FC SAN omrežje ima dve Qlogic in dve Emulex FC kartici, vsaka od FC kartic pa ima po dve vtičnici. Emulex FC kartici sta namenjeni za priklop na ostale strežnike, ki jim strežnik TS7650G nudi virtualne tračne knjižnice. Qlogic FC kartici sta namenjeni za priklop strežnika TS7650G na diskovna polja, kjer ima ta strežnik repozitorij. Priklop strežnika TS7650G na ostale strežnike in na diskovna polja je izveden preko FC SAN omrežja, kar pa si bomo bolj detajlno ogledali v naslednjem poglavju.

Ko je strežnik TS7650G konfiguriran, lahko na njem definiramo virtualne tračne knjižnice z virtualnimi tračnimi pogoni, virtualnimi tračnimi kasetami, virtualnimi roboti in virtualnimi policami, na katerih so virtualne tračne kasete. Ko so virtualne tračne knjižnice definirane, ostali strežniki do njih dostopajo enako kot do ne-virtualiziranih tračnih knjižnic [1].

2.2 Fibre Channel Storage Area Network (FC SAN)

Združenje Storage Network Industry Association (SNIA) definira SAN omrežje, kot omrežje katerega namen je prenos podatkov med računalniškimi sistemi in pomnilnimi elementi. SAN omrežje sestoji iz komunikacijske infrastrukture, ki zagotavlja fizične povezave in upravljalsko plastjo, ki organizira povezave, pomnilne elemente in računalniške sisteme tako, da je prenos podatkov varen in robusten. FC SAN omrežje je SAN omrežje, ki povezuje elemente preko FC protokola. FC protokol v SAN omrežju je, podobno kot TCP/IP protokol, sestavljen iz večih plasti, ki skrbijo za povezljivost, varnost in robustnost.

Enostavno FC SAN omrežje deluje tako, da se računalniški sistemi in pomnilni elementi med seboj ne vidijo, četudi so priklopljeni na isto FC SAN omrežje vse dokler ne naredimo con, znotraj katerih definiramo katere vtičnice računalniških sistemov in pomnilnih elementov bodo medsebojno vidne. Ko so cone definirane, so pomnilniški elementi na voljo računalniškim sistemom za bralno-pisalne operacije [2, 6].

2.3 San Volume Controller (SVC)

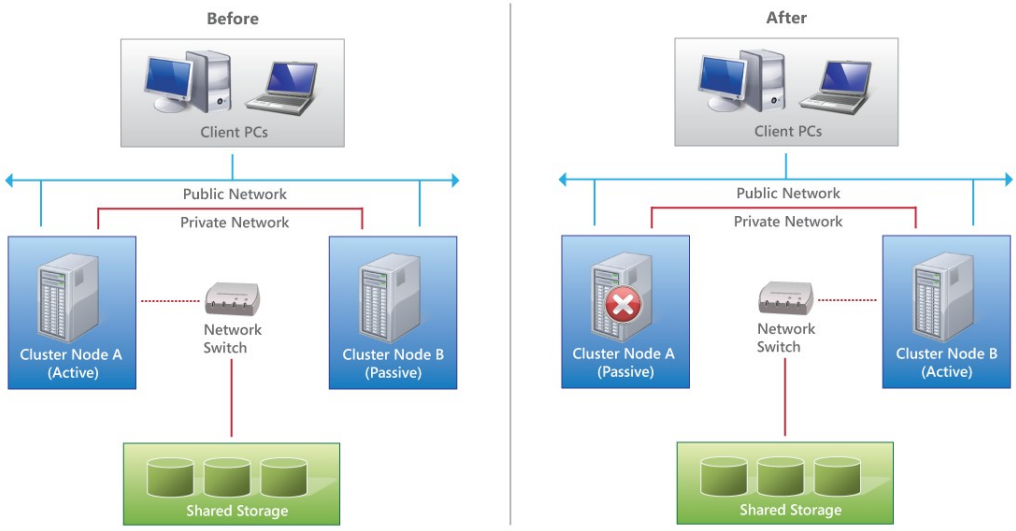
SVC je element v FC SAN omrežjih, ki predstavlja dodaten nivo virtualizacije diskovnih polj in vedno deluje v gruči, ki je sestavljena iz vsaj dveh vozlišč. Za vsako vozlišče je uporabljen strežnik IBM System X, na katerem se izvaja Linux jedro in posebno programsko okolje za virtualizacijo. Za priklop na FC SAN omrežje ima vsako vozlišče štiri vtičnice, preko katerih teče komunikacija med vozlišči v gruči, kot tudi prenos podatkov med SVC vozlišči in strežniki, ter SVC vozlišči in diskovnimi polji. Zaradi zaščite podatkov v primeru izpada napajanja je vsako vozlišče SVC priklopljeno na napajanje UPS. SVC deluje tako, da se diskovnim poljem predstavlja kot strežnik, strežnikom pa kot diskovno polje. Diskovna polja, ki so ji na voljo, združi v skupine, ki jih potem razdeli na virtualne diske. Te virtualne diske potem ponudi strežnikom za vhodno-izhodne operacije. Na ta način SVC razporedi vhodno-izhodne operacije na več diskovnih polj s čimer pridobimo na zmogljivosti, istočasno pa izgubimo na zanesljivosti, saj izpad enega diskovnega polja pomeni izpad vseh virtualnih diskov, ki so v skupini, v kateri je to diskovno polje. Do tega sicer naj ne bi prišlo, saj mora že samo diskovno polje zagotavljati dovolj veliko zanesljivost. SVC omogoča še vrsto drugih uporabnih funkcionalnosti, kot so sinhrona in asinhrona replikacija na daljavo (ang.

Metro/Global Mirror), zrcaljenje virtualnih diskov in tako dalje. Uporabili smo jo zaradi funkcionalnosti zrcaljenja virtualnih diskov, s čimer smo nadomestili replikacijo na nivoju vozlišč TS7650G in jo prestavili na nivo repozitorija, oziroma virtualnih diskov, na katerih je repozitorij [3].

2.4 Windows Server Failover Clustering

Windows Server Failover Clustering (WSFC) je Microsoftova rešitev za zagotavljanje visoke razpoložljivosti aplikacij, servisov in podatkov. Microsoft je prvi WSFC, oziroma Microsoft Cluster Services (MSCS), kot se je takrat ta rešitev imenovala, predstavil z operacijskim sistemom Windows NT 4.0 Enterprise Edition. Od takrat se je ta rešitev precej spremenila, predvsem pa izboljšala. WSFC je v primerjavi z MSCS tudi precej poenostavljen, tako da uporabniku za konfiguriranje gruče ni potrebno detajlno poznavanje njenega delovanja.

WSFC je skupina samostojnih strežnikov oziroma vozlišč, ki so fizično povezana v isto omrežje in programsko povezani preko WSFC programske opreme. Skupino vozlišč, ki se nahaja v isti domeni, upravljamo kot enoten sistem. Ponavadi so v skupno omrežje povezana preko večih povezav, ravnotako pa so vozlišča ponavadi povezana v SAN omrežje v katerem si delijo iste vire. WSFC deluje tako da premika vire med vozlišči. Premik vira iz enega vozlišča na drugega se sproži v primeru, ko pride do okvare, ki prekine delovanje ali pa dosegljivost tega vira. Premik lahko sprožimo tudi ročno v primeru, da bi na trenutno aktivnem vozlišču radi izvedli vzdrževalna dela. Uporabnik premika vira iz enega vozlišča na drugega ponavadi niti ne opazi, ravno tako pa do njega dostopa na enak način, ne glede na to na katerem vozlišču se trenutno nahaja. Na sliki 3 je prikazano delovanje WSFC, ki poskrbi za preklop na vozlišče B, v primeru izpada vozlišča A [4, 7].



Slika 3. Windows Server Failover Cluster [7].

2.5 Perl in distribucija ActivePerl

ActivePerl je brezplačna Perl distribucija, ki deluje na Windows, Linux in Mac OS X sistemih. Namenjena je razvijalcem odprtokodnih projektov, zato vključuje samo podporo na forumih [5].

Perl je visoko-nivojski skriptni programski jezik, ki ga je leta 1987 razvil Larry Wall. Veliko funkcionalnosti je Perl prevzel iz ostalih programskih jezikov kot so C, shell script, AWK in sed (ang. stream editor). Perl je zelo dober pri razčlenjevanju teksta, zato je v devetdesetih pridobil na popularnosti kot CGI skriptni jezik. Uporablja se tudi za razvoj grafičnih aplikacij, sistemsko administracijo, programiranje mrežnih aplikacij, v financah, v bioinformatiki in ostalih aplikacijah [8].

2.6 PsTools

PsTools je skupek orodij za poenostavitev administracije Windows sistemov [9]. Zajema naslednja orodja:

- PsExec – izvede proces na daljavo,
- PsFile – pokaže datoteke, ki so odprte na daljavo,
- PsGetSid - pokaže SID (ang. Security ID) računalnika ali pa uporabnika,
- PsInfo - izpiše podatke o sistemu,

- PsKill - ustavi oziroma ubije proces,
- PsList - izpiše detajlne informacije o procesu,
- PsLoggedOn - izpiše listo uporabnikov, ki so prijavljeni lokalno, ali preko deljenja virov,
- PsLogList - izpiše zapise v dnevniku dogodkov,
- PsPasswd - spremeni geslo uporabnika,
- PsService - je namenjen pregledovanju in kontroliranju servisov,
- PsShutdown - izklopi ali ponovno zažene računalnik,
- PsSuspend - suspendira proces,
- PsUptime - prikaže koliko časa od zadnjega zagona sistem deluje.

V našem primeru smo uporabili samo orodje PsExec, s katerim smo v kontekstu lokalnega sistemskega uporabnika sprejeli gostiteljske ključe vseh sistemov na katere se skripte Perl povezujejo preko SSH protokola. To smo morali narediti, ker WSFC gruča izvaja vse svoje aktivnosti v kontekstu lokalnega sistemskega uporabnika, kar pa posledično pomeni, da se izvajajo tudi vse skripte v kontekstu lokalnega sistemskega uporabnika in ravno tako tudi vsi ukazi, ki jih izvedejo skripte.

3 UPORABA TS7650G V GRUČI Z REPLIKACIJO

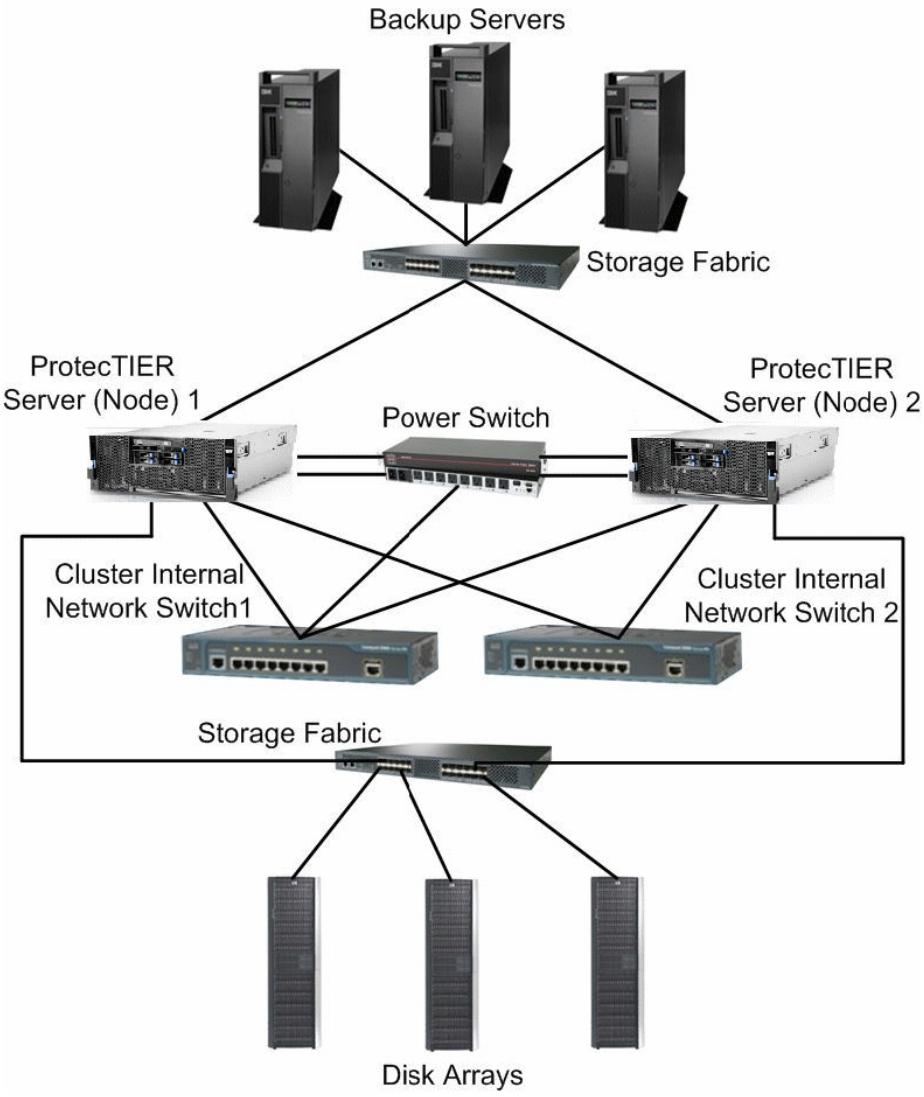
V primeru potrebe po zagotavljanju višje razpoložljivosti sistema nam IBM omogoča, da dve vozlišči TS7650G povežemo v gručo. V tej strukturi lahko primarno vozlišče TS7650G podatke kopira na sekundarno vozlišče TS7650G z uporabo replikacije. Če želimo omogočiti replikacijo, moramo sistem že na začetku postaviti in konfigurirati z mislijo na to. Razlika v fizični postavitni je v tem, da moramo obe vozlišči TS7650G priklopiti na napajanje preko mrežnega PDU (ang. Power Distribution Unit) elementa, poleg tega pa ju moramo priklopiti tudi na dve gigabitni mrežni stikali, preko katerih poteka komunikacija med njima.

Mrežni PDU je element, v katerega pripeljemo dve neodvisni veji napajanja, kar pomeni, da morata biti tudi na različnih varovalkah. Na izhodu tega elementa so vtičnice, kamor priklopimo porabnike, v našem primeru obe vozlišči TS7650G. Vtičnice so deljene na dve skupini in sicer na prvo, ki je priklopljena na eno vejo napajanja in drugo, ki je priklopljena na drugo vejo napajanja. Ko priklapljamo vozlišče TS7650G, moramo paziti da vsakega od obeh napajalnikov priklopimo v svojo skupino vtičnic. S tem dosežemo to, da je vsako vozlišče TS7650G priklopljeno na dva neodvisna vira napajanja, kar dodatno poveča zanesljivost delovanja. Smiselnost priklopa preko mrežnega PDU elementa pa je v tem, da v primeru da se eno vozlišče TS7650G preneha odzivati, lahko drugo prvemu izklopi in nato ponovno vklopi napajanje. Ko se prvo vozlišče izklopi in ponovno zažene je odpravljena možnost, da bi obe vozlišči hkrati spreminjali skupne podatke, pri čemer bi lahko prišlo do okvare podatkov. Fizični priklop gruče TS7650G z replikacijo je prikazan na sliki 4.

Komunikacija med vozliščema TS7650G, ki teče preko dveh gigabitnih mrežnih stikal, je namenjena predvsem replikaciji podatkov na virtualnih tračnih kasetah. Izvaja se po deduplikaciji, zato je količina podatkov, ki jo je potrebno prenesti, temu primerno manjša. Replikacijo je možno konfigurirati na več načinov, kar omogoča uporabnikom, da definirajo več različnih politik po katerih poteka replikacija med vozliščema TS7650G. Politike replikacije lahko uporabnik določi za posamezno virtualno tračno kaseto, skupino virtualnih tračnih kaset ali pa kar celotno virtualno tračno knjižnico. Replikacija se izvaja asinhrono na nivoju posamezne virtualne tračne kasete, trenutno stanje replikacije pa si lahko ogledamo preko uporabniškega vmesnika. Po končani replikaciji vozlišče TS7650G na ciljni strani naredi validacijo virtualne tračne kasete, s čimer je zagotovljena integriteta prenesenih

podatkov. Šele po uspešni validaciji je virtualna tračna kaseta na voljo za uporabo [1].

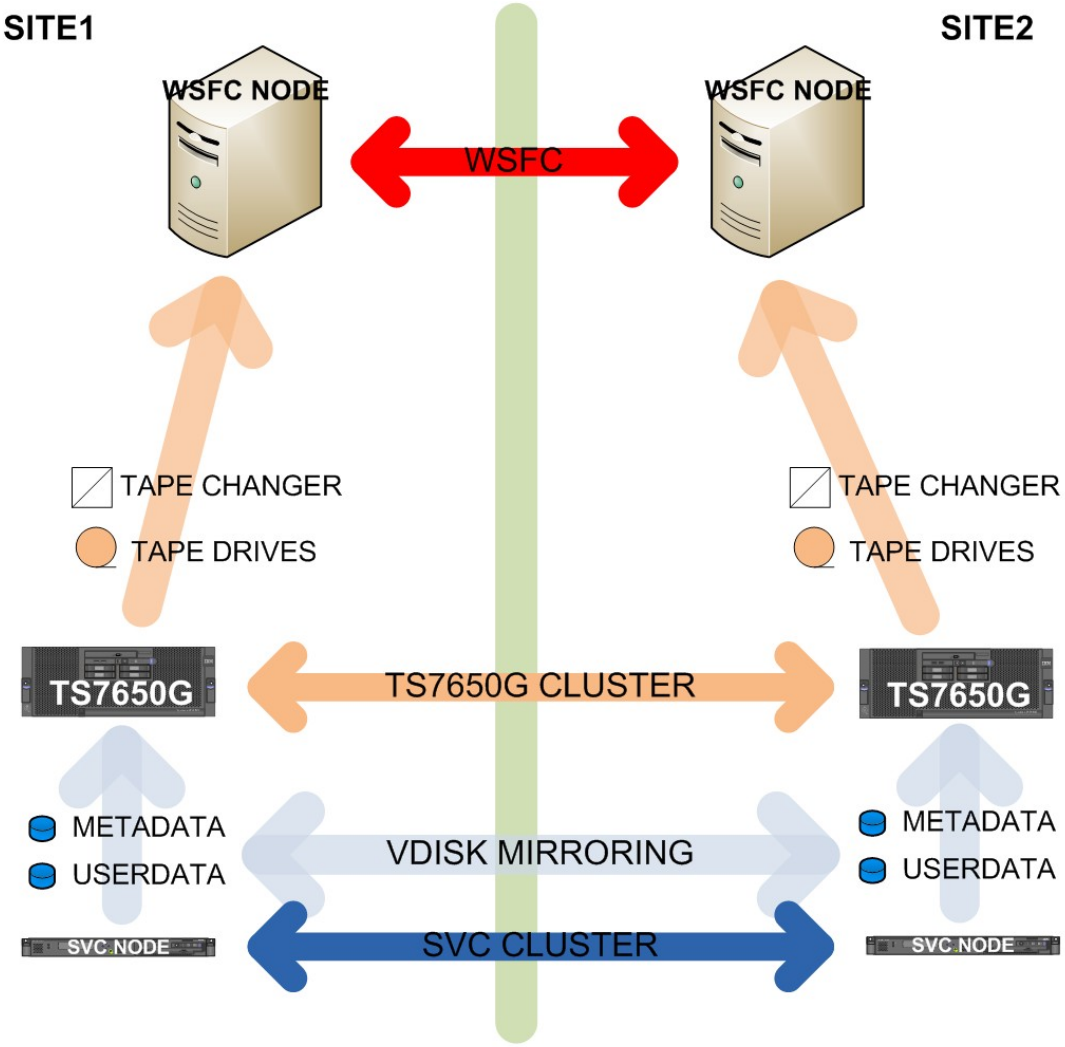
V veliko primerih sta primarna in sekundarna lokacija računskega centra v različni zgradbi, ali pa vsaj različnem prostoru, kar pa predstavlja problem. Gruča TS7650G z replikacijo je omejena s tem, da morata biti vozlišči relativno blizu, saj sta priklopljeni na isti mrežni PDU element. V tem primeru gruča TS7650G z replikacijo ni možna, zato je potrebno to rešiti na drug način.



Slika 4. TS7650G gruča [1].

4 IMPLEMENTACIJA VOZLIŠČ TS7650G V GRUČI WSFC

Kot smo videli v prejšnjem poglavju IBMova rešitev za postavitev sistema TS7650G v gručo z replikacijo za zagotavljanje visoke razpoložljivosti ni vedno najboljša rešitev, zato si bomo v tem poglavju ogledali, kako smo visoko razpoložljivost dosegli z implementacijo vozlišč TS7650G v gruči WSFC. Glavna razlika med obema rešitvama je v tem, da naša rešitev, ki je prikazana na sliki 5, ne uporablja replikacije na nivoju tračnih kaset, pač pa zrcaljenje na nivoju virtualnih diskov, kar pomeni, da je repozitorij na obeh straneh isti, za razliko od IBMove rešitve, pri kateri je repozitorij različen (vsako vozlišče TS7650G ima svoj repozitorij), enake pa so samo tiste tračne kasete, ki jih vključimo v replikacijo. Ker naša rešitev ne vključuje replikacije, avtomatično odpade potreba po dveh mrežnih stikalih ("Cluster Internal Network Switch 1" in "Cluster Internal Network Switch 2" na sliki 4). Druga razlika med obema rešitvama pa je ta, da IBMova rešitev za preprečevanje okvare podatkov repozitorija v primeru tako izmenovane "split-brain" situacije uporablja mrežni PDU element za ponovni zagon vozlišča TS7650G, naša rešitev pa preprečuje okvaro padatkov repozitorija z izklopom vtičnic na FC stikalih, s čimer fizično prepreči dostop do repozitorija. Posledično pri naši rešitvi odpade potreba po mrežnem PDU elementu ("Power Switch" na sliki 4), dodatno pa si moramo zagotoviti dostop do vseh FC stikal, na katere sta priklopljeni vozlišči TS7650G in pravice za upravljanje z vtičnicami, na katere sta priklopljeni ti dve vozlišči.



Slika 5. Implementacija TS7650G v gruči WSFC.

4.1 Priprava vozlišč v gruči WSFC

Za implementacijo vozlišč TS7650G v gruči WSFC je potrebno najprej na obe vozlišči WSFC namestiti skriptno okolje ActivePerl, klienta SSH PuTTY in orodja PsTools. Ko končamo z namestitvijo, moramo nanju prenesti tudi vse skripte, dodati nov generični skriptni vir in ga konfigurirati. Ker je naš generični skriptni vir precej okoren v primerjavi z ostalimi viri v gruči pride do določenih problemov. Gruča WSFC namreč predpostavlja, da je neka normalna časovna omejitev za prehod med stanji vira v gruči, 3 oziroma 5 minut, vendar v našem primeru lahko prehod med stanji traja tudi 15 minut in več. Zaradi tega je potrebno nastavitvi

"pendingtimeout" in "deadlocktimeout" v gruči WSFC spremeniti na 30 minut. Za povezovanje preko SSH protokola na FC stikala in vozlišči TS7650G, bomo na obeh virih v gruči WSFC potrebovali tudi javni in zasebni ključ.

Za zagotovitev zgoraj opisanih zahtev smo izvedli spodaj navedene korake v podanem zaporedju:

- namestimo ActivePerl, PuTTY in PsTools na obe vozlišči WSFC,
- vse skripte prenesemo na lokalni disk na obe vozlišči WSFC,
- v skripto VTL.vbs vnesemo nastavitve, ki odgovarjajo našemu okolju,
- v aplikaciji Failover Cluster Manager dodamo nov generični skriptni vir,
- v nastavitvah za gručo WSFC popravimo "pendingtimeout" in "deadlocktimeout" za vse vire, ki so odvisni od našega generičnega skriptnega vira, vključno z njim,
- na obeh vozliščih WSFC generiramo javni in zasebni ključ z uporabo orodja PuTTYgen.

4.2 Priprava vozlišč TS7650G

Po končani pripravi vozlišč WSFC je potrebno pripraviti obe vozlišči TS7650G tako, da bosta vozliščema WSFC, oba prikazala identične tračne robote in tračne pogone. To dosežemo s tako imenovanim postopkom za zamenjavo vozlišča. To je postopek, ki še nekonfigurirano vozlišče TS7650G konfigurira na podlagi repozitorija nekega drugega vozlišča TS7650G. Konkretno v našem primeru to pomeni, da najprej na SVCju pripravimo repozitorij, konfiguriramo prvo vozlišče TS7650G, potem ga izklopimo, mu odvzamemo repozitorij, repozitorij pokažemo drugemu vozlišču TS7650G in poženemo postopek za zamenjavo vozlišča. Ko je postopek končan, imamo praktično dve identični vozlišči TS7650G, med katerima lahko ročno preklapljamo. Da vozlišči pripravimo tudi na avtomatizirano preklapljanje preko generičnega skriptnega vira v gruči WSFC, pa je potrebno narediti še nekaj sprememb. Na vozliščih je potrebno narediti novega uporabnika in profil zanj. Preko tega uporabnika bo generični skriptni vir potem preverjal delovanje vozlišča TS7650G. Ker se

bo generični skriptni vir na obe vozlišči TS7650G povezoval preko SSH protokola z uporabo asimetrične enkripcije, moramo na obe vozlišči TS7650G distribuirati javna ključa obeh vozlišč WSFC. Ker si bosta sistema izmenjevala IP naslov, je potrebno uskladiti tudi gostiteljske ključe in sicer tako da imata obe vozlišči TS7650G enake. Generični skriptni vir pri preklopu med vozliščema TS7650G zamenja tudi IP naslov, kar pa bi prekinilo SSH sejo, zato smo to zamenjavo izvedli z uporabo enostavne skripte. To skripto je potrebno prenesti na obe vozlišči TS7650G. Pri preklopu med vozliščema TS7650G generični skriptni vir preveri tudi lokacijo trenutno aktivnega vozlišča TS7650G. Ker sta sistema identična, smo za ta namen na obeh vozliščih naredili tudi posebno datoteko, ki hrani to informacijo. V našem primeru se na sistem prijavljamo z uporabnikom root, za katerega je potrebno ročno omogočiti možnost prijave preko SSH protokola, saj je v osnovi ta možnost onemogočena. Na koncu je potrebno na obeh vozliščih TS7650G uskladiti naslove, oziroma WWPNje (ang. World Wide Port Number) prednjih FC vtičnic.

Za zagotovitev zgoraj opisanih zahtev smo izvedli spodaj navedene korake v naslednjem zaporedju:

- pripravimo repozitorij na SVCju za obe vozlišči TS7650G,
- namestimo in konfiguriramo prvo vozlišče TS7650G,
- kreiramo novega uporabnika na prvem vozlišču TS7650G,
- kreiramo profil za novega uporabnika na prvem vozlišču TS7650G,
- konfiguriramo drugo vozlišče TS7650G na podlagi obstoječega repozitorija,
- kreiramo profil za novega uporabnika na drugem vozlišču TS7650G,
- kopiramo gostiteljske ključe iz prvega vozlišča TS7650G na drugega,
- kopiramo VTLstandby skripto na obe vozlišči TS7650G,
- kreiramo datoteko z informacijo o lokaciji na obeh vozliščih TS7650G,
- distribuiramo javna ključa obeh vozlišč WSFC na obe vozlišči TS7650G,
- omogočimo prijavo root uporabnika preko SSH povezave,
- spremenimo WWPNje na drugem vozlišču TS7650G.

4.3 Priprava FC stikal

Ko sta obe vozlišči TS7650G pripravljeni, je potrebno pripraviti FC stikala, na katera sta priključeni obe vozlišči TS7650G. Ker gruča WSFC deluje tako, da izklaplja in vklaplja vtičnice na FC stikalih, ki so povezani na vozlišči TS7650G, moramo gruči WSFC zagotoviti dostop do njih. Uporabnikom na FC stikalih, ki jih bo gruča uporabljala, moramo dodeliti dovolj pravic za upravljanje z vtičnicami.

Dostop do FC stikal je v našem primeru zagotovljen preko SSH protokola. Ker je pri avtentikaciji z geslom vsakič potrebno interaktivno vnesti geslo, je ta izvedena z asimetrično avtentikacijo z uporabo javnega in zasebnega ključa. Uporabnik, ki ga gruča WSFC uporablja za upravljanje z vtičnicami, je "vtluser" in sicer na vseh štirih FC stikalih. Obe vozlišči WSFC imata svoj par ključev, tako da je na vseh štirih FC stikalih omogočena asimetrična avtentikacija preko ključev za uporabnika "vtluser" in na vsako FC stikalo sta distribuirana javna ključa obeh vozlišč WSFC.

Ker so na teh štirih stikalih priključeni tudi drugi sistemi in vozlišča, je uporabnik "vtluser" omejen izključno na operacije nad vtičnicami, ki so priključene na vozlišči TS7650G. Ta omejitev je izvedena z uporabo tako imenovanih administrativnih domen (ang. Administrative Domain). Administrativne domene so v bistvu skupine, ki vsebujejo vtičnice, stikala in naprave, vsakemu uporaniku pa lahko določimo s katerimi administrativnimi domenami lahko upravlja. Uporabnik s tem, ko je omejen na samo določene administrativne domene, ne more upravljati s stikali, vtičnicami in napravami, ki niso znotraj teh administrativnih domen. Uporabnika "vtluser" moramo zato omejiti samo na eno administrativno domeno, ki vsebuje vtičnice, ki so povezane na obe vozlišči TS7650G, ne sme pa vsebovati nobenega stikala ali naprave, kar pomeni da ne more spreminjati konfiguracije stikala. To storimo z naslednjimi koraki:

- na obeh SAN omrežjih naredimo novo administrativno domeno, ki vsebuje samo vtičnice, ki so priključene na vozlišči TS7650G,
- na vsakem od štirih FC stikal naredimo novega uporabnika "vtluser" in mu dodelimo zgoraj omenjeno administrativno domeno,
- na vseh štirih FC stikalih omogočimo uporabniku "vtluser" prijavo preko SSH protokola,

- na vsa štiri stikala distribuiramo javna ključa obeh vozlišč WSFC preko SSH protokola.

4.4 Shranjevanje SSH gostiteljskih ključev

Na koncu je potrebno na obeh vozliščih WSFC še sprejeti in shraniti gostiteljske ključe vseh štirih FC stikal in obeh vozlišč TS7650G. To storimo z uporabo PsTools orodja, ki nam omogoča izvajanje ukazov v kontekstu lokalnega sistemskega uporabnika. Ker imata obe vozlišči TS7650G skupaj tri IP naslove, vsak svojega in enega skupnega, je potrebno gostiteljske ključe sprejeti za vse tri IP naslove. Gostiteljski ključ za skupni IP naslov lahko sprejmemo takrat, ko je aktiven na prvem ali drugem vozlišču TS7650G, saj so gostiteljski ključi na obeh vozliščih enaki. SSH gostiteljske ključe shranimo z naslednjimi koraki:

- z uporabo orodja PsExec se prijavimo v sistem z lokalnim sistemskim uporabnikom in vzpostavimo SSH povezavo na vsako FC stikalo,
- z uporabo orodja PsExec se prijavimo v sistem z lokalnim sistemskim uporabnikom in vzpostavimo SSH povezavo na vozlišče TS7650G, ki je trenutno v stanju delovanja,
- z uporabo orodja PsExec se prijavimo v sistem z lokalnim sistemskim uporanikom in vzpostavimo SSH povezavo na vozlišče TS7650G, ki je trenutno v stanju pripravljenosti,
- vozlišče TS7650G, ki je trenutno v stanju delovanja, prestavimo v stanje pripravljenosti, drugo vozlišče TS7650G pa iz stanja pripravljenosti prestavimo v stanje delovanja,
- z uporabo orodja PsExec se prijavimo v sistem z lokalnim sistemskim uporabnikom in vzpostavimo SSH povezavo na vozlišče TS7650G, ki je trenutno v stanju pripravljenosti.

5 DELOVANJE TS7650G VOZLIŠČ V GRUČI WSFC

Implementacija obeh vozlišč TS7650G je izvedena tako, da sta v gruči WSFC prikazani kot generični skriptni vir (ang. Generic Script Resource), kar pomeni, da gruča WSFC z njim operira enako kot z vsakim drugim virom, ki mu je dodeljen. Generični skriptni vir je v bistvu skripta, ki ima implementirane točno določene funkcije, ki jih gruča WSFC kliče, ko upravlja s tem virom. V našem primeru skripta algoritmov nima implementiranih neposredno v funkcijah, pač pa kliče zunanje skripte Perl, ki opravijo potrebne korake, potem pa vrnejo kodo, preko katere sporočijo uspešnost ali pa neuspešnost izvedenih korakov. S tem smo dosegli, da je vozlišči TS7650G relativno enostavno implementirati tudi na drugih sistemih, saj je skriptni jezik Perl veliko bolj razširjen kot pa Visual Basic Script, ki je praktično omejen samo na Microsoftova okolja.

5.1 WSFC generični skriptni vir

Generični skriptni vir (ang. Generic Script Resource) v kontekstu gruče WSFC je vir, katerega logika je zapisana v skripti. Skripta je lahko napisana v skriptnih jezikih JavaScript ali pa Visual Basic Script, mi smo izbrali slednjega. Za pravilno delovanje je skripto potrebno napisati v točno določeni obliki kot prikazuje slika 6.

```
'   ... Insert your script-level global variables and definitions here
'   ... e.g. Resource.LogInformation("ScriptWide Global Stuff is Run")
'   ... Code placed here is outside any entry point function.
'   ... It is run once when the script is created
'   ... and once when the script is placed online.

Function Open( )
'   ... Insert your Open code here.
End Function

Function Online( )
'   ... Insert your Online code here.
'   ... Online is executed once when the resource is placed online.
End Function

Function LooksAlive( )
'   ... Insert your LooksAlive code here.
'   ... LooksALive is executed at specified intervals.
End Function

Function IsAlive( )
'   ... Insert your IsAlive code here.
'   ... IsAlive is executed at specified intervals
'   ... or when a LooksAlive call fails.
End Function

Function Offline( )
'   ... Insert your Offline code here.
'   ... Offline is executed once when the resource is placed offline.
End Function

Function Close( )
'   ... Insert your Close code here.
End Function

Function Terminate( )
'   ... Insert your Terminate code here.
'   ... Terminate is executed once when the script terminates.
End Function
```

Slika 6. Generični skriptni vir.

Iz slike 6 je razvidna struktura kode. Koda, ki je izven zgoraj omenjenih sedmih funkcij se izvede pri kreiranju instance generičnega skriptnega vira in pri postavitni le tega v stanje delovanja (ang. Online state), pravtako pa so tu definirane globalne spremenljivke. V našem primeru smo izven funkcij definirali samo globalne spremenljivke, vsa logika pa je implementirana znotraj funkcij oziroma v Perl skriptah, ki jih funkcije kličejo.

Ko gruča WSFC pokliče funkcijo Open, naredi novo instanco generičnega skriptnega vira. V tej fazi se izvede koda, ki je izven funkcij in koda, ki je znotraj funkcije Open. Namen Open funkcije je pripraviti generični skriptni vir na začetek uporabe.

Funkcijo Online gruča WSFC pokliče, ko ročno v aplikaciji Failover Cluster Manager prestavimo generični skriptni vir iz stanja pripravljenosti v stanje delovanja ali pa, ko gruča WSFC sama prestavi generični skriptni vir iz stanja pripravljenosti v stanje delovanja, naprimer pri sprožitvi postopka preklopa na drugo vozlišče WSFC. V tej fazi se izvede koda, ki je znotraj funkcije Online.

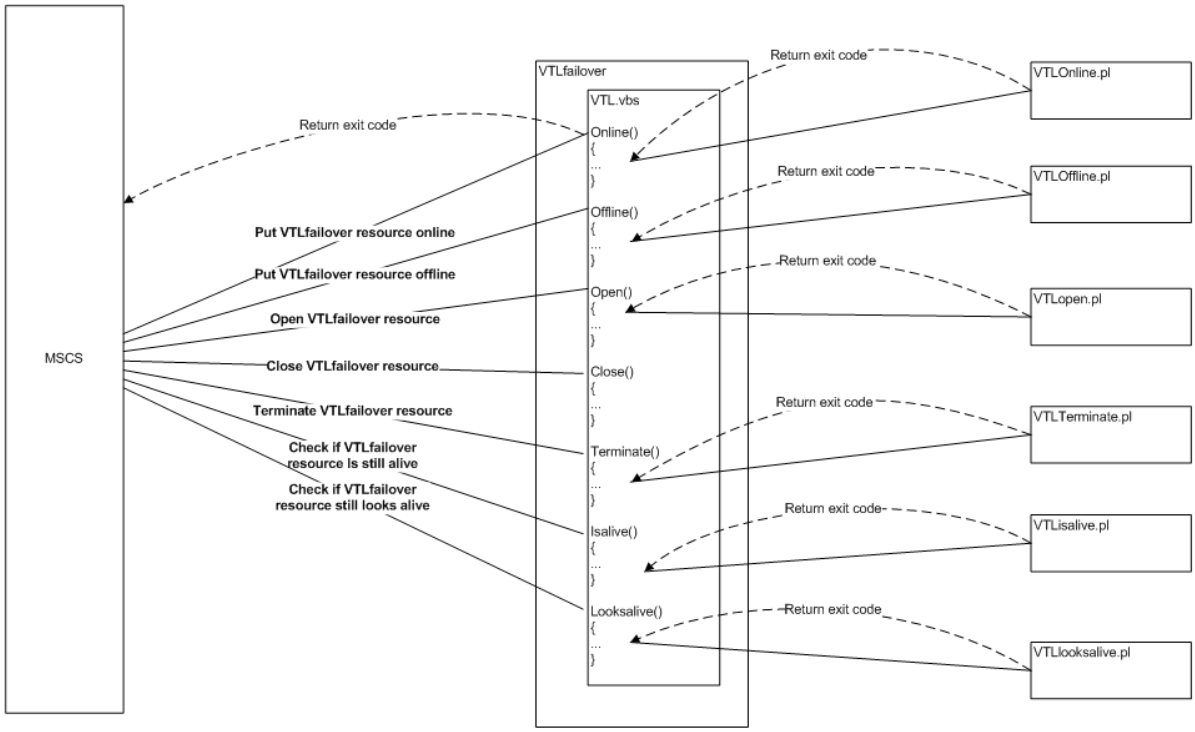
Funkciji LooksAlive in IsAlive gruča WSFC kliče periodično takrat, ko je generični skriptni vir v stanju delovanja. Namen obeh funkcij je preveriti ali je generični skriptni vir še vedno odziven oziroma delujoč. V tej fazi se izvede koda, ki je znotraj LooksAlive oziroma IsAlive funkcije. Razlika med obema funkcijama je v tem da LooksAlive funkcija izvede relativno hiter pregled, IsAlive funkcija pa izvede bolj poglobljen pregled in je tudi časovno bolj potratna. Ker je funkcija IsAlive bolj časovno potratna v primerjavi s funkcijo LooksAlive, se posledično izvaja manj pogosto. Periodo izvajanja obeh funkcij se lahko spremeni.

Funkcijo Offline gruča WSFC pokliče, ko ročno v aplikaciji Failover Cluster manager prestavimo generični skriptni vir iz stanja delovanja v stanje pripravljenosti, ali pa, ko gruča WSFC sama prestavi generični skriptni vir iz stanja delovanja v stanje pripravljenosti, naprimer pri sprožitvi postopka preklopa na drugo vozlišče WSFC. V tej fazi se izvede koda, ki je znotraj funkcije Offline. Namen Offline funkcije je spraviti generični skriptni vir v stanje pripravljenosti, tako da ne opravlja svoje funkcionalnosti, je pa pripravljen za vklop oziroma prehod v stanje delovanja.

Ko gruča WSFC pokliče funkcijo Close, izbriše obstoječo instanco generičnega skriptnega vira. V tej fazi se izvede koda, ki je znotraj funkcije Close. Namen Close funkcije je sprostiti nadzor gruče WSFC nad generičnim skriptnim virom.

Funkcijo Terminate gruča WSFC pokliče, ko je potrebno generični skriptni vir na silo ustaviti, naprimer ko se funkcija IsAlive ne izvede uspešno, kar pomeni, da je generični skriptni vir najverjetneje neodziven ali pa ne deluje pravilno. Gruča pokliče to funkcijo tudi v primeru, ko se Offline funkcija ne izvede uspešno. Takoj ko se Terminate funkcija prične izvajati, gruča WSFC označi generični skriptni vir kot nedosegljiv, kar v primeru odvisnosti ostalih virov od tega vira sproži tudi postopke za postavitev virov v stanje pripravljenosti in pa postopek preklopa na drugo vozlišče WSFC, če je to seveda nastavljeno. V tej fazi se izvede koda, ki je znotraj funkcije Terminate. Namen Terminate funkcije je prekiniti vse aktivnosti, ki jih je ta generični skriptni vir izvajal.

Slika 7 prikazuje povezavo med gručo WSFC, generičnim skriptnim virom in Perl skriptami, ki jih generični skriptni vir kliče.



Slika 7. WSFC.

5.2 Skripte Perl

V tem poglavju si bomo ogledali najprej funkcionalnosti skript Perl za normalno upravljanje z vsemi štirimi FC stikali in obema vozliščema TS7650G, potem pa še algoritme oziroma logiko posameznih skript, na podlagi katere lahko generičnemu skriptnemu viru zagotovijo uspešno dokončanje naloge, ki jo ta od posamezne skripte Perl zahteva.

5.2.1 Funkcionalnosti skript Perl

Skripte Perl uporabljajo za svoje delovanje različna orodja in funkcionalnosti, kot so program *plink*, za vzpostavljanje SSH povezave na obe vozlišči TS7650G in vsa štiri FC stikala, program *eventcreate* za pisanje dogodkov v dnevnik operacijskega sistema Windows, pisanje vseh pomembnih korakov v poseben dnevnik, zaganjanje ukazov na FC stikalih za vklapljanje in izklapljanje vtičnic, izvajanje ukazov na vozliščih TS7650G za prehode med stanjem

delovanja in stanjem pripravljenosti, ter izvajanje ukazov za nadziranje stanja vozlišča TS7650G ko je ta v stanju delovanja.

5.2.1.1 Aplikacija PuTTY in ukaz plink

PuTTY [10] je odprtokodna aplikacija, ki znotraj grafičnega vmesnika emulira terminal. Podpira mrežne protokole SSH, Telnet, rlogin in pa povezavo neposredno preko serijskega vmesnika. Aplikacija je bila v osnovi napisana za okolje Windows, vendar je bila sčasoma prenesena tudi na druge operacijske sisteme. PuTTY je grafična aplikacija in je taka namenjena interakciji uporabnika. V našem primeru pa SSH povezave vzpostavlja skripta Perl, zato smo uporabili kar program *plink*, ki ga uporablja tudi aplikacija PuTTY. *Plink* je program, ki ga zaženemo v ukazni vrstici in mu podamo nastavitve v obliki argumentov. Na sliki 8 lahko vidimo primer njegove uporabe za preverjanje statusa vozlišča TS7650G. V tem primeru smo podali naslednje argumente:

- -l root - uporabniško ime, ki ga uporabimo za povezavo na vozlišče,
- -i c:\ibm\vtl\key.ppk - zasebni ključ, s katerim se avtenticiramo,
- 172.32.201.22 – IP naslov vozlišča na katerega se prijavljamo,
- /opt/dtc/ptcli/ptcli Libraries --login ptcluster --force – ukaz, ki ga izvedemo na oddaljenem vozlišču.

```
C:\>plink -l root -i c:\ibm\vtl\key.ppk 172.32.201.22 /opt/dtc/ptcli/ptcli Libraries --login ptcluster --force
<?xml version="1.0" encoding="UTF-8"?>
<response command="Libraries" host-name="VTL1" host-time="1304344872"
  host-time-string="02-May-2011 16:01" local-time="1304344872"
  local-time-string="02-May-2011 16:01" status="success">
  <libraries-list num-returned="1">
    <library name="virtual_3584" unique-id="2199023256586"/>
  </libraries-list>
</response>

C:\>_
```

Slika 8. Uporaba programa *plink*.

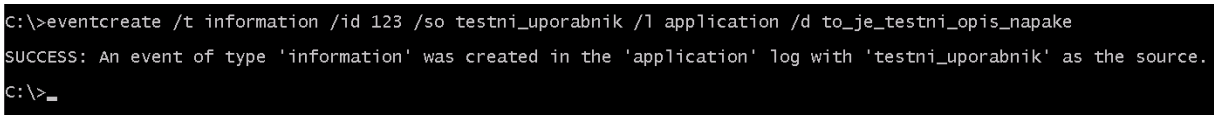
Plink smo uporabili za izvajanje ukazov na obeh vozliščih TS7650G in na vseh štirih FC stikalih.

5.2.1.2 Program eventcreate in pisanje v aplikacijski dnevnik

Eventcreate je program, ki je že priložen operacijskemu sistemu Windows 2008 in je

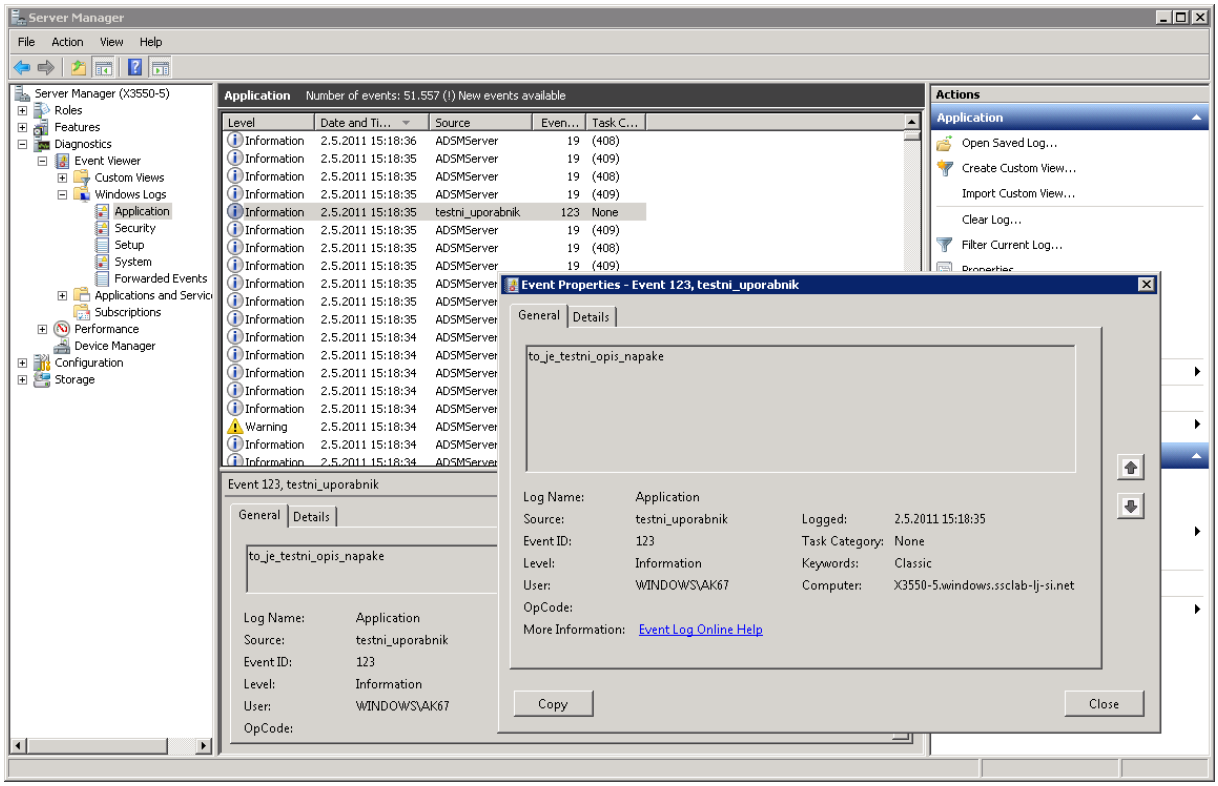
namenjen pisanju dogodkov v dnevnik operacijskega sistema Windows. Zaženemo ga v ukazni vrstici, podatke pa mu posredujemo v obliki argumentov. Na sliki 9 je prikazan primer uporabe ukaza *eventcreate* in sicer z naslednjimi argumenti:

- /t information – pove da gre za informativni dogodek,
- /id 123 – pove da gre za dogodek z identifikacijsko številko 123,
- /so testni_uporabnik – pove kdo je vir dogodka oziroma kdo je zapisal dogodek v dnevnik,
- /l application – pove, da se mora dogodek zapisati v aplikacijski dnevnik,
- /d to_je_testni_opis_napake – dejanski opis napake, ki se bo zapisal v dnevnik.



Slika 9. Uporaba programa *eventcreate*.

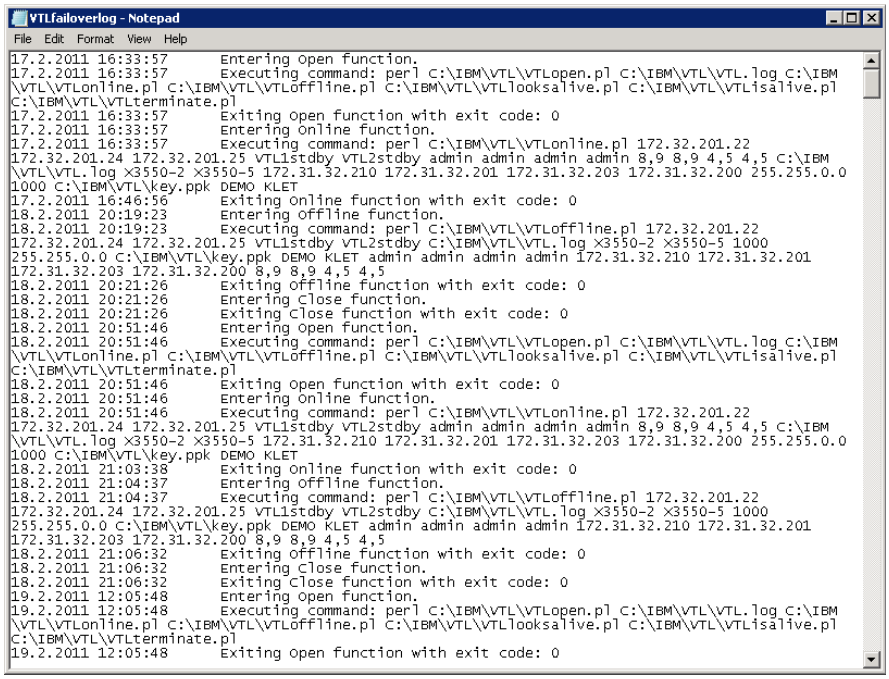
Po izvedbi ukaza se v aplikacijskem devniku pojavi nov dogodek (slika 10). *Eventcreate* smo uporabili za pisanje dogodkov v aplikacijski dnevnik in sicer v primeru, da je potrebna intervencija skrbnika sistema. V primeru manjših nepravilnosti, ki ne vplivajo na delovanje sistema pa dogodkov ne pišemo v aplikacijski dnevnik, pač pa samo v dnevnik generičnega skriptnega vira.



Slika 10. Nov dogodek v aplikacijskem dnevniku.

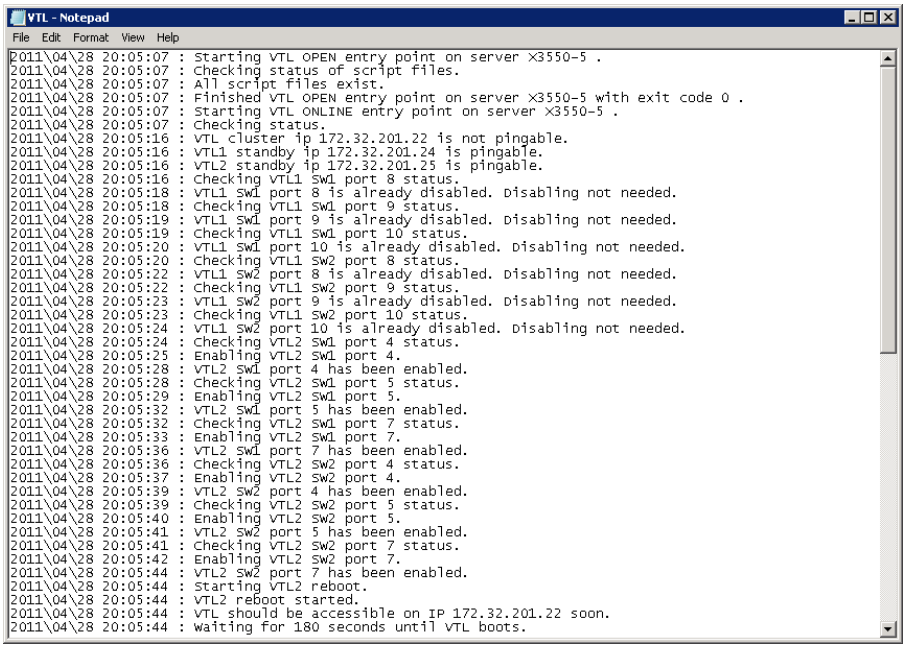
5.2.1.3 Pisanje v dnevnik generičnega skriptnega vira

Vsi pomembni koraki, ki jih opravi generični skriptni vir in pa skripte Perl, ki jih kliče, se zapisujejo v dva različna dnevnika. V prvega zapisuje samo generični skriptni vir in sicer informacije o tem katera funkcija se izvaja, katero Perl skripto pokliče in s kakšno kodo se zaključi izvajanje poklicane Perl skripte. Vsak zapis ima na začetku tudi informacijo o uri in datumu. Primer tega dnevnika je prikazan na sliki 11.



Slika 11. Dnevnik generičnega skriptnega vira.

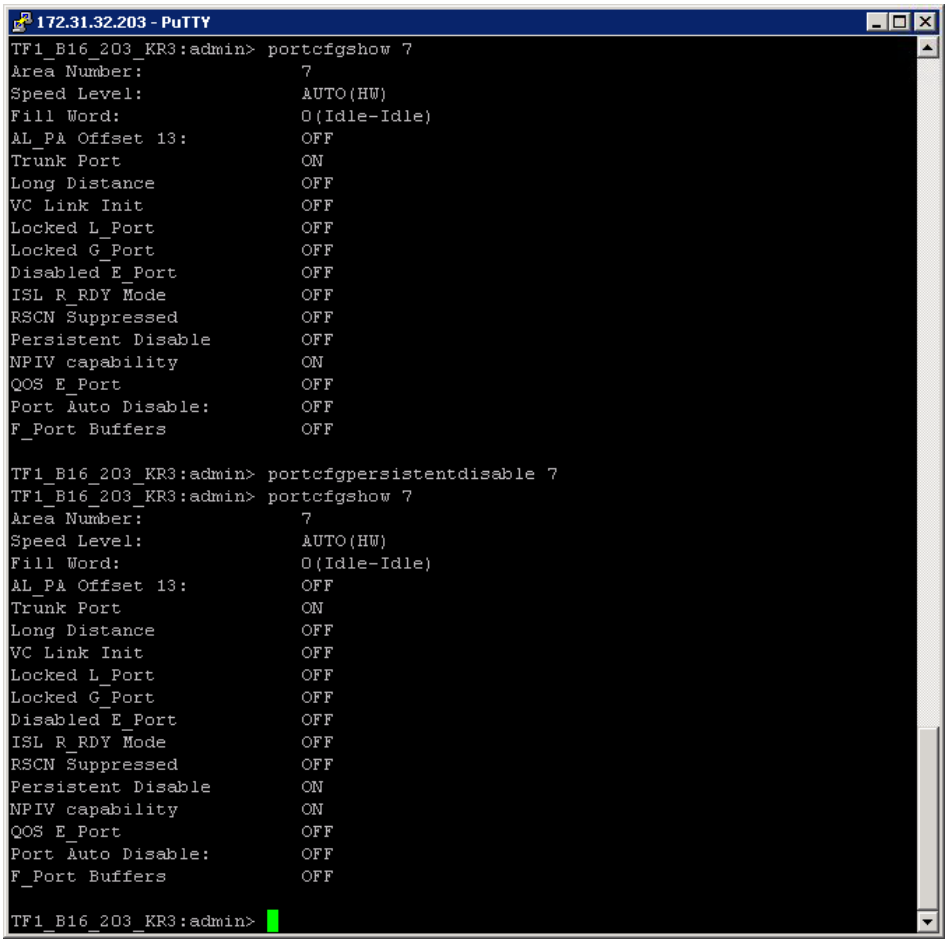
V drugi dnevnik zapisujejo samo skripte Perl. Ob začetku izvajanja katerekoli od skript Perl, se v dnevnik zapiše katera skripta Perl se je začela izvajati in na katerem od vozlišč WSFC se izvaja. Podobno se zgodi ob zaključku izvajanja, ko se v dnevnik zapiše tudi s katero kodo se je izvajanje zaključilo. Poleg tega pa v dnevnik zapisuje vsak korak, ki ga opravi. Izjema sta samo Perl skripti VTLisalive.pl in VTLlooksalive.pl, ki v dnevnik pišeta samo v primeru, ko se neuspešno zaključita. Razlog je v tem, da bi ti dve skripti lahko zelo hitro napolnili dnevnik, saj se izvajata periodično in poleg tega še precej pogosto. Primer tega dnevnika je prikazan na sliki 12.



Slika 12. Dnevnik skript Perl.

5.2.1.4 Upravljanje vtičnic FC stikal

Pri prestavljanju vozlišč TS7650G iz stanja pripravljenosti v stanje delovanja in obratno, morajo skripte Perl poskrbeti tudi za vklop in izklop vtičnic na FC stikalih. To storijo tako, da z uporabo PuTTY ukaza *plink* vzpostavijo SSH povezavo na vsakega od štirih FC stikal in izvedejo ukaz za vklop ali izklop vtičnice. Za vklop izvedejo ukaz „portcfgpersistenenable“, ki mu sledi številka vtičnice, za izklop pa ukaz „portcfgpersistentdisable“, ki mu pravtako sledi številka vtičnice. Po končanemu ukazu za vklop ali izklop preverijo še uspešnost z ukazom „portcfgshow“, ki mu sledi številka vtičnice. Na sliki 13 je prikazan primer izklopa vtičnice 7 in pa preverjanje uspešnosti. Če dobro pogledamo je pred izklopom vrstica „Persistent Disable“ nastavljena na „OFF“, kar pomeni da je vtičnica vklopljena, po izklopu pa je nastavljena na „ON“, kar pomeni da je vtičnica izklopljena. Analogno naredimo pri vklopu vtičnice, le da uporabimo ukaz „portcfgpersistentenable“ in da mora biti po izvedenemu ukazu vrstica „Persistent Disable“ nastavljena na „OFF“.



Slika 13. Izklop vtičnice.

5.2.1.5 Upravljanje z vozliščema TS7650G

Perl skripte za upravljanje z vozliščema TS7650G uporabljajo SSH protokol in sicer z uporabo ukaza *plink*. Za postavitev vozlišča TS7650G iz stanja pripravljenosti v stanje delovanja, ga morajo skripte ponovno zagnati, potem pa počakati, da se ponoven zagon vključno s servisi uspešno zaključi. Za ponovni zagon skripte Perl uporabijo Linux ukaz „shutdown -r -F now“ s čimer sporočijo vozlišču naj se ponovno zažene, pri tem pa naredi tudi pregled lokalnih datotečnih sistemov. Ko je vozlišče TS7650G v fazi ponovnega zagona, skripte z uporabo ukaza „ping“ periodično preverjajo, če je IP naslov gruče TS7650G že dosegljiv. Ko postane dosegljiv, skripte počakajo 5 sekund da se sshd servis, na katerem teče SSH strežnik, zažene. Ko se sshd servis zažene, začnejo skripte periodično preverjati stanje servisa *vtfd* na vozlišču TS7650G z ukazom „service vtfd status“, vse dokler se servis dokončno ne zažene. Za postavitev vozlišča TS7650G iz stanja delovanja v stanje

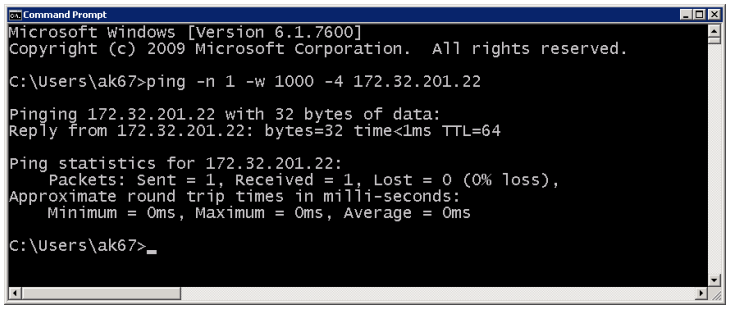
pripravljenosti, morajo skripte najprej izklopiti vse servise z ukazi „service vtfd stop“, „service gfs stop“, „service clvmd stop“ in „service cman stop“. Te ukaze morajo skripte izvesti v točno takšnem zaporednju, poleg tega pa morajo za preverjanje uspešnosti izvesti tudi ukaze „service vtfd status“, „service gfs status“, „service clvmd status“ in „service cman status“. Ko so vsi štirje servisi izklopljeni, morajo skripte spremeniti še gostiteljsko ime (ang. Hostname) in IP naslov vozlišča TS7650G. To naredijo tako, da z ukazom „(>&/dev/null /root/VTLstandby $vtlstdbyhostname $ip $netmask & exit)“ zaženemo skripto /root/VTLstandby, ki ima v argumentih podano gostiteljsko ime, IP naslov in masko podomrežja. Zgoraj omenjen ukaz je prirejen temu, da lahko IP naslov spremenimo tudi preko SSH protokola, saj prekine sejo še pred spreminjanjem IP naslova. Če bi enostavno uporabili Linux ukaz „ifconfig“ preko SSH protokola bi si s tem odžagali vejo, saj bi prekinili SSH sejo, pri čemer pa bi program *plink* obvisel. V tem primeru bi morali počakati toliko časa, da bi prišlo do prekinitve seje zaradi neodzivnosti, kar pa bi samo dodatno povečalo čas potreben za dokončanje izvajanja skript Perl. Po zamenjavi IP naslova in gostiteljskega imena skripte preverijo še uspešnost in sicer z ukazom „ping“ za preverjanje IP naslova in z Linux ukazom „hostname“ na vozlišču TS7650G, za preverjanje gostiteljskega imena.

5.2.1.6 Nadziranje vozlišča TS7650G v stanju delovanja

Za preverjanje delovanja vozlišča TS7650G, ki je v stanju delovanja, skripte uporabljajo dva pristopa:

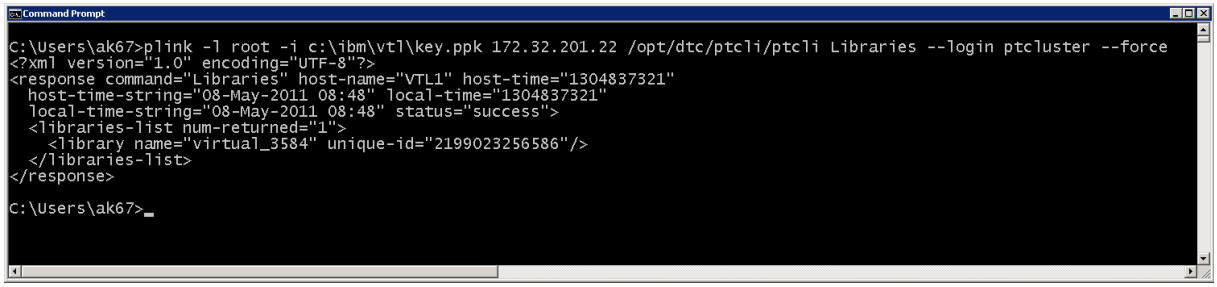
- uporaba enostavnega ukaza ping
- izpis virtualnih tračnih knjižnic na vozlišču TS7650G.

Pri preverjanju z ukazom ping skripte izvedejo ukaz „ping -n 1 -w 1000 -4 172.32.201.22“ in počakajo na rezultat. Argument „-n 1“ pomeni, da bo ukaz poslal samo eno ICMP ECHO zahtevo. Argument „-w 1000“ pomeni, da bo ukaz na ICMP ECHO odgovor čakal eno sekundo. Argument „-4“ pomeni da bo ukaz poslal ICMP ECHO zahtevo po standardu IP verzije 4. Na sliki 14 lahko vidimo primer uporabe ukaza ping. Skripte po končanju izvajanja ukaza preverijo izpis ukaza. Če izpis vsebuje niza „Received = 1“ in „Approximate round trip times in milli-seconds:“ skripte smatrajo, da je ukaz uspešen oziroma, da ja ICMP ECHO odgovor prišel nazaj.



Slika 14. Ukaz ping.

Pri preverjanju z izpisom virtualnih tračnih knjižnic na vozlišču TS7650G skripte izvedejo ukaz „/opt/dtc/ptcli/ptcli Libraries --login ptcluster –force“ na vozlišču TS7650G z uporabo ukaza *plink* in počakajo na rezultat. Argument „--login ptcluster“ pomeni, da se bo ukaz logiral z uporabnikom, ki smo ga določili pri kreiranju profila ptcluster. Argument „–force“ pomeni, da bo ukaz izvedel izpis virtualnih knjižnic tudi v primeru, da je uporabnik že prijavljen oziroma trenutno izvaja ukaz. V tem primeru bo prijavljenega uporabnika odjavil. Na sliki 15 lahko vidimo primer zagona ukaza ptcli. Skripte po končanju izvajanja ukaza preverijo izpis ukaza. Če izpis vsebuje niz „status="success"“ skripte smatrajo, da je ukaz uspešen oziroma, da vozlišče TS7650G deluje.



Slika 15. Ukaz ptcli.

5.2.2 Algoritmi posameznih Perl skript

V naslednjih poglavjih si bomo ogledali kako deluje vsaka od šestih Perl skript, ki upravljajo neposredno z vozliščema TS7650G in vsemi štirimi FC stikali.

5.2.2.1 Online

VTLonline.pl je skripta, ki je odgovorna za to, da zagotovi, da je vozlišče TS7650G na drugi lokaciji v stanju pripravljenosti, na tej lokaciji pa v stanju delovanja. To doseže tako, da najprej preveri dejansko stanje in izvede temu primerne korake. V tabeli 1 so prikazana možna stanja na katera lahko naleti.

SCENARIJ	IP TS7650G GRUČE	IP PRVEGA TS7650G VOZLIŠČA	IP DRUGEGA TS7650G VOZLIŠČA	LOKACIJA AKTIVNEGA TS7650G VOZLIŠČA	LOKACIJA AKTIVNEGA WSFC VOZLIŠČA
1	AKTIVEN	AKTIVEN	NEAKTIVEN	2	1
2	AKTIVEN	AKTIVEN	NEAKTIVEN	2	2
3	AKTIVEN	NEAKTIVEN	AKTIVEN	1	2
4	AKTIVEN	NEAKTIVEN	AKTIVEN	1	1
5	AKTIVEN	NEAKTIVEN	NEAKTIVEN	1	1
6	AKTIVEN	NEAKTIVEN	NEAKTIVEN	1	2
7	AKTIVEN	NEAKTIVEN	NEAKTIVEN	2	1
8	AKTIVEN	NEAKTIVEN	NEAKTIVEN	2	2
9	NEAKTIVEN	AKTIVEN	AKTIVEN	NIKJER	1
10	NEAKTIVEN	AKTIVEN	AKTIVEN	NIKJER	2
11	NEAKTIVEN	NEAKTIVEN	AKTIVEN	NIKJER	1
12	NEAKTIVEN	AKTIVEN	NEAKTIVEN	NIKJER	1
13	NEAKTIVEN	NEAKTIVEN	AKTIVEN	NIKJER	2
14	NEAKTIVEN	AKTIVEN	NEAKTIVEN	NIKJER	2
15	NEAKTIVEN	NEAKTIVEN	NEAKTIVEN	NIKJER	1
16	NEAKTIVEN	NEAKTIVEN	NEAKTIVEN	NIKJER	2

Tabela 1. Možna stanja.

Koraki, ki jih skripta opravi, za vsako od naštetih stanj so sledeči (zaporedna številka predstavlja stanje):

1. Skripta postavi prvo vozlišče TS7650G, ki je trenutno v stanju delovanja, v stanje pripravljenosti. Ko je prvo vozlišče TS7650G v stanju pripravljenosti, prestavi drugo vozlišče TS7650G v stanje delovanja.
2. V tem primeru je stanje že takšno kot mora biti. Skripta v tem primeru samo preveri,

ali drugo vozlišče TS7650G deluje.

3. Skripta postavi drugo vozlišče TS7650G, ki je trenutno v stanju delovanja, v stanje pripravljenosti. Ko je drugo vozlišče TS7650G v stanju pripravljenosti, prestavi prvo vozlišče TS7650G v stanje delovanja.
4. V tem primeru je stanje že tako kot mora biti. Skripta v tem primeru samo preveri, ali drugo vozlišče TS7650G deluje.
5. V tem primeru skripta ravna enako kot v stanju 4.
6. V tem primeru skripta ne more drugega vozlišča TS7650G prestaviti v stanje delovanja, saj ni dosegljiv. V tem primeru skripta ustvari not dogodek v aplikacijskem dnevniku operacijskega sistema Windows in vrne kodo 1, kar pomeni da izvajanje ni bilo uspešno.
7. V tem primeru skripta ne more prvega vozlišča TS7650G prestaviti v stanje delovanja, saj ni dosegljiv, zato ravna enako kot v stanju 6.
8. V tem primeru skripta ravna enako kot v stanju 4.
9. Skripta postavi prvo vozlišče TS7650G iz stanja pripravljenosti v stanje delovanja.
10. Skripta postavi drugo vozlišče TS7650G iz stanja pripravljenosti v stanje delovanja.
11. Skripta ravna enako kot v primeru 7.
12. V tem primeru skripta ravna enako kot v stanju 9.
13. V tem primeru skripta ravna enako kot v stanju 10.
14. V tem primeru skripta ravna enako kot v stanju 6.
15. V tem primeru skripta ravna enako kot v stanju 6.
16. V tem primeru skripta ravna enako kot v stanju 6.

Poleg v tabeli 1 naštetih stanj so sicer teoretično možna tudi ostala stanja, ampak ker do njih praktično skoraj ni mogoče priti smo jih zanemarili. V primeru, da bi do katerega od teh stanj vseeno prišlo, bi skripta ravnala tako kot je opisano v stanju 6. Če v času izvajanja skripta naleti na težavo, ki bi ogrozila delovanje sistema, ali pa ne more izvesti katerega od korakov, ki so nujno potrebni za normalno delovanje gruče TS7650G, prekine delovanje in vrne kodo 1, kar pomeni da se izvajanje ni zaključilo uspešno. V primeru da se skripta izvede brez težav, vrne kodo 0, kar pomeni, da se je izvajanje uspešno zaključilo.

5.2.2.2 Offline

VTLoffline.pl je skripta, ki je odgovorna za to, da vozlišče TS7650G, ki je na tej lokaciji, postavi v stanje pripravljenosti. To doseže tako, da najprej preveri na katerem vozlišču WSFC je bila pognana, potem pa preveri na kateri lokaciji je vozlišče TS7650G, ki je trenutno v stanju delovanja. Če je vozlišče TS7650G, ki je v stanju delovanja, na isti lokaciji kot vozlišče WSFC, na katerem je bila pognana ta skripta, potem postavi to vozlišče TS7650G v stanje pripravljenosti. V primeru, da v času izvajanja skripta naleti na težavo, ki bi ogrozila delovanje sistema, ali pa ne more izvesti katerega od korakov, ki so nujno potrebni za normalno delovanje gruče TS7650G, prekine delovanje in vrne kodo 1, kar pomeni da se izvajanje ni uspešno zaključilo. V primeru, da se skripta izvede brez težav, vrne kodo 0, kar pomeni, da se je izvajanje uspešno zaključilo.

5.2.2.3 Terminate

VTLterminate.pl je skripta, ki je odgovorna za to, da vozlišče TS7650G, ki je na tej lokaciji, postavi na silo v stanje pripravljenosti. To doseže tako, da najprej preveri na katerem vozlišču WSFC je bila pognana, potem pa preveri na kateri lokaciji je vozlišče TS7650G, ki je trenutno v stanju delovanja. Če je vozlišče TS7650G, ki je v stanju delovanja, na isti lokaciji kot vozlišče WSFC, na katerem je bila pognana ta skripta, potem postavi to vozlišče TS7650G v stanje pripravljenosti. Razlika med to skripto in skripto VTLoffline.pl je ta, da ta skripta nadaljuje izvajanje ne glede na to ali vmes pride do kakšnega neuspešnega koraka. Ta skripta vedno vrne kodo 0, ker pomeni da se je delovanje uspešno zaključilo.

5.2.2.4 Open

VTLopen.pl je skripta, ki je odgovorna za to, da preveri ali je generični skriptni vir sploh možno pripraviti na začetek uporabe. To izvede tako, da preveri če so vse ostale skripte na voljo in če so, potem skripta vrne kodo 0, kar pomeni, da se je izvajanje uspešno končalo. V primeru, da skripta VTLopen.pl ugotovi, da katera od ostalih skript manjka, smatra, da upravljanje z vozliščema TS7650G ni izvedljivo, zato vrne kodo 1, kar pomeni, da se je izvajanje končalo neuspešno.

5.2.2.5 Isalive

VTLisalive.pl je skripta ki je odgovorna za to, da preveri ali je vozlišče TS7650G na tej lokaciji aktivno in delujoče. To izvede tako da se preko SSH povezave poveže nanj in izvede enostaven ukaz za izpis virtualnih knjižnic. Če dobi uspešen odgovor, to smatra kot zadosten dokaz, da je TS7650G node aktiven in delujoč. V tem primeru skripta vrne kodo 0, kar pomeni, da se je izvajanje uspešno zaključilo. V primeru, da skripta dobi neuspešen odgovor ali ne dobi odgovora, smatra da vozlišče TS7650G na tej lokaciji ne deluje, zato vrne kodo 1.

5.2.2.6 Looksalive

VTLlooksalive.pl je skripta, ki je odgovorna za to, da na hitro preveri ali je vozlišče TS7650G na tej strani aktivno. To izvede tako, da enostavno z uporabo ICMP ECHO zahteve preveri ali je IP naslov gruče TS7650G trenutno aktiven. V primeru da skripta dobi ICMP ECHO odgovor, smatra, da trenutno aktivno vozlišče TS7650G deluje, zato vrne kodo 0. V primeru da skripta ne dobi ICMP ECHO odgovora, smatra, da trenutno aktivno vozlišče TS7650G ne deluje, zato vrne kodo 1.

6 DELOVANJE IN TESTIRANJE

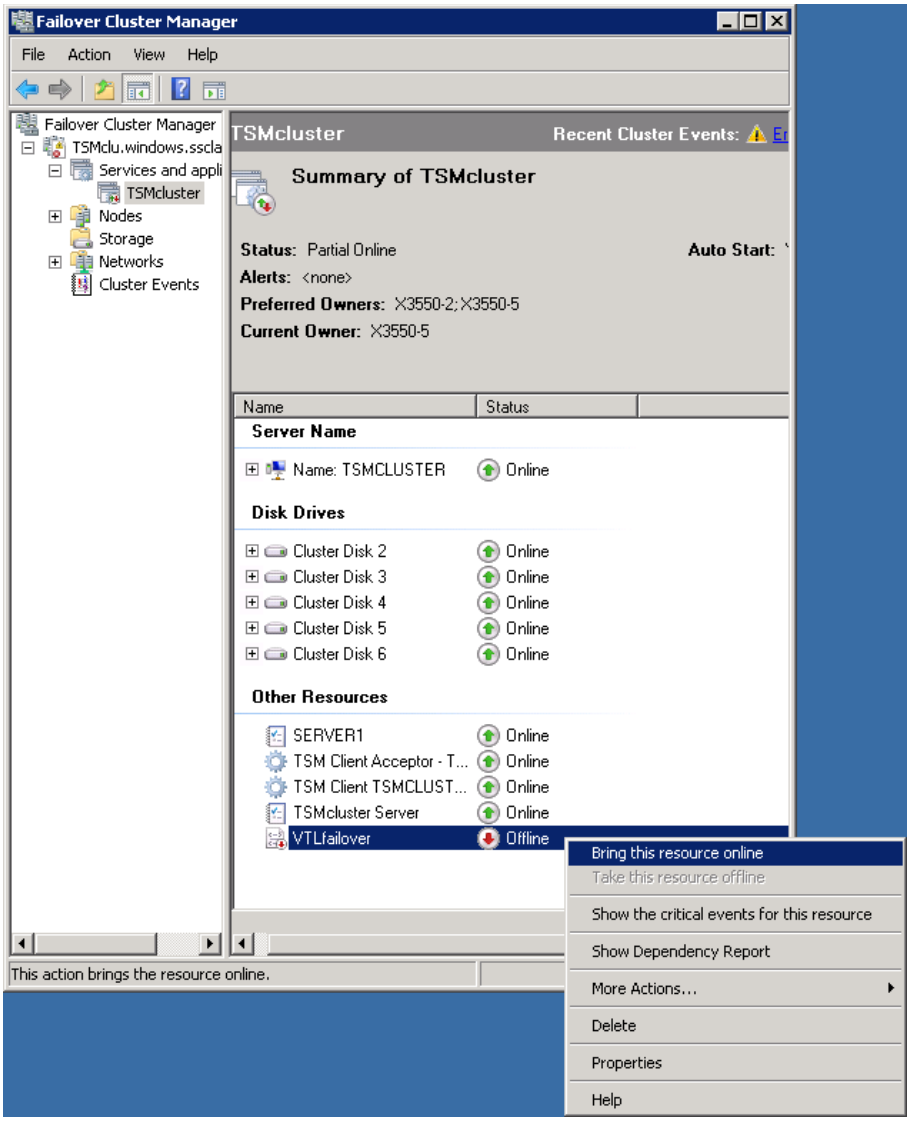
Ko imamo obe vozlišči TS7650G implementirani v gruči WSFC, si lahko pogledamo delovanje na primerih. Na začetku preden sistem zaženemo prvič, moramo zagotoviti, da sta obe vozlišči TS7650G v stanju pripravljenosti. Ko sta vozlišči v stanju pripravljenosti, lahko v aplikaciji Failover Cluster Manager postavimo generični skriptni vir v stanje delovanja. V tem poglavju si bomo na primeru pogledali kako postaviti generični skriptni vir v stanje delovanja in kako generični skriptni vir v primeru napake trenutno aktivnega vozlišča TS7650G aktivira drugo vozlišče TS7650G.

6.1 Prestavljanje generičnega skriptnega vira v stanje delovanja

Pri pripravi vozlišč TS7650G za implementacijo v gruči WSFC smo primarno vozlišče TS7650G izklopili, sekundarno pa je v stanju delovanja, zato moramo sekundarno vozlišče TS7650G ročno postaviti v stanje pripravljenosti, primarno vozlišče TS7650G postaviti v stanje delovanja in potem še v stanje pripravljenosti. To storimo z naslednjimi koraki:

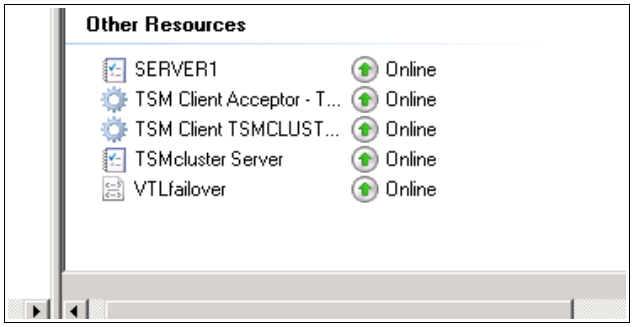
- prijavimo se na sekundarno vozlišče TS7650G in izklopimo servise *vtfd, gfs, clvmd* in *cman* v točno takem vrstnem redu,
- po izklopu servisov izklopimo vse vtičnice na FC stikalih, ki so povezane na sekundarno vozlišče TS7650G,
- na koncu zamenjamo še IP naslov in gostiteljsko ime sekundarnega vozlišča TS7650G,
- na FC stikalih vklopimo vse vtičnice, ki so povezane na primarno vozlišče TS7650G,
- vklopimo primarno vozlišče TS7650G,
- ko se primarno vozlišče TS7650G dokončno zažene, se prijavimo nanj in izklopimo servise *vtfd, gfs, clvmd* in *cman* v podanem vrstnem redu,
- po izklopu servisov izklopimo vse vtičnice na FC stikalih, ki so povezane na primarno vozlišče TS7650G,
- na koncu zamenjamo še IP naslov in gostiteljsko ime primarnega vozlišča TS7650G.

Sedaj ko sta obe vozlišči TS7650G v stanju pripravljenosti, lahko prestavimo generični skriptni vir v stanje delovanja kot prikazuje slika 16.



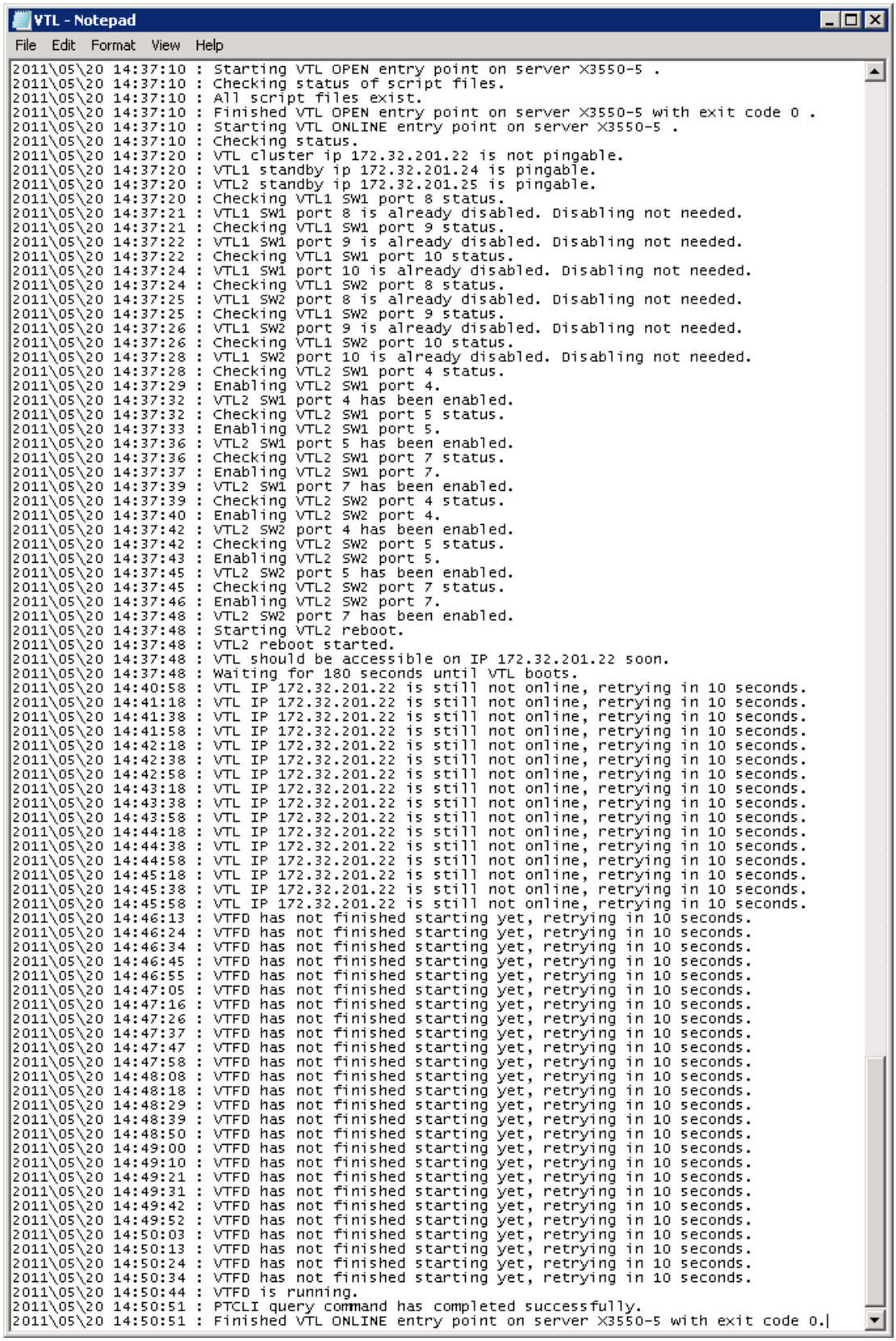
Slika 16. Postavljanje generičnega skriptnega vira v stanje delovanja.

Po približno 15 minutah generični skriptni vir preide v stanje delovanja kot kaže slika 17.



Slika 17. Generični skriptni vir v stanju delovanja.

Na sliki 18 si lahko ogledamo korake, ki jih je naredil generični skriptni vir. Kot vidimo iz zapisov v dnevnik, je generični skriptni vir najprej preveril stanje ostalih skript Perl. Za tem je generični skriptni vir preveril stanje obeh vozlišč TS7650G na podlagi tega, kateri IP naslovi so trenutno aktivni. Ko je ugotovil, da sta obe vozlišči TS7650G v stanju pripravljenosti, je nadaljeval z vtičnicami na FC stikalih. Ker so bile vse vtičnice na FC stikalih, ki so priklopljene na primarno vozlišče TS7650G, izklopljene, jih ni izklapljal. Vtičnice, ki so priklopljene na sekundarno vozlišče TS7650G so bile izklopljene, zato jih je vklopil. Na koncu je le še ponovno zagnal sekundarno vozlišče TS7650G in počakal da se je postavilo in da je servis *vtfd* končal z izvajanjem.



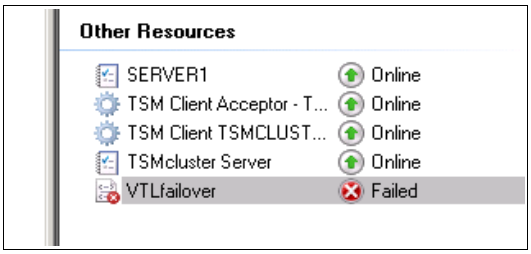
Slika 18. Dnevnik postavljanja generičnega skriptnega vira v stanje delovanja.

6.2 Selitev generičnega skriptnega vira v primeru napake

Za testiranje smo napako simulirali tako, da smo na trenutno aktivnem vozlišču TS7650G izklopili mrežno vtičnico, preko katere generični skriptni vir preverja stanje tega vozlišča. Ker vozlišče TS7650G ni bilo dosegljivo, je generični skriptni vir smatral, da ne deluje, zato je aktiviral drugo vozlišče TS7650G.

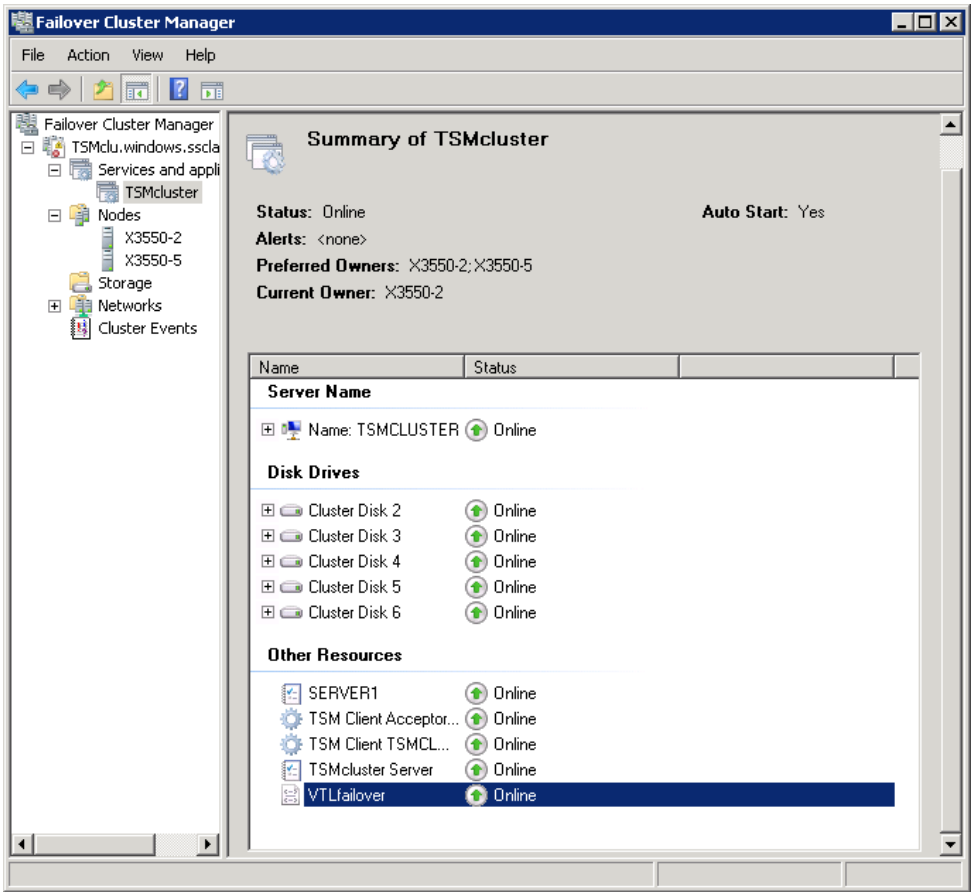
Stanje pred simulacijo napake je bilo tako, da je bilo sekundarno vozlišče TS7650G v stanju delovanja, primarno vozlišče TS7650G pa v stanju pripravljenosti. Generični skriptni vir je bil v stanju delovanja na sekundarnem strežniku.

Točno ob 15:57:00 uri smo izklopili mrežno vtičnico, kar je generični skriptni vir zaznal in se postavil v stanje napake kot je razvidno na sliki 19.



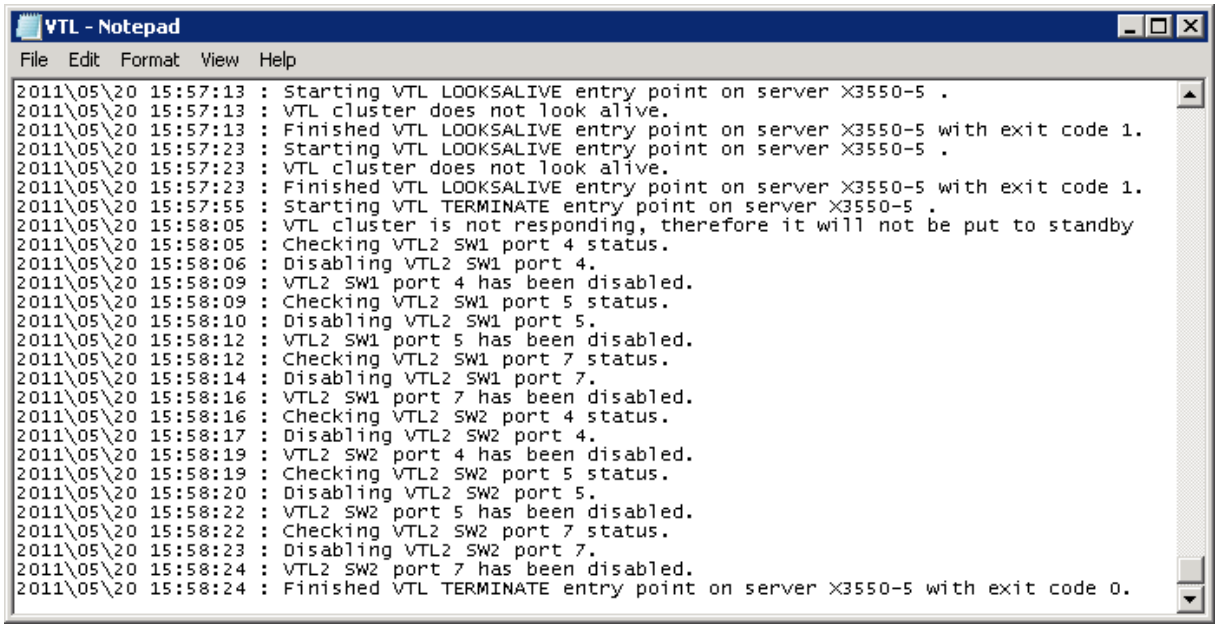
Slika 19. Generični skriptni vir v stanju napake.

Kmalu po tem, ko je generični skriptni vir zaznal napako, je začel s postopkom preklopa na drugo vozlišče TS7650G in ga uspešno zaključil, kot je vidno na sliki 20.



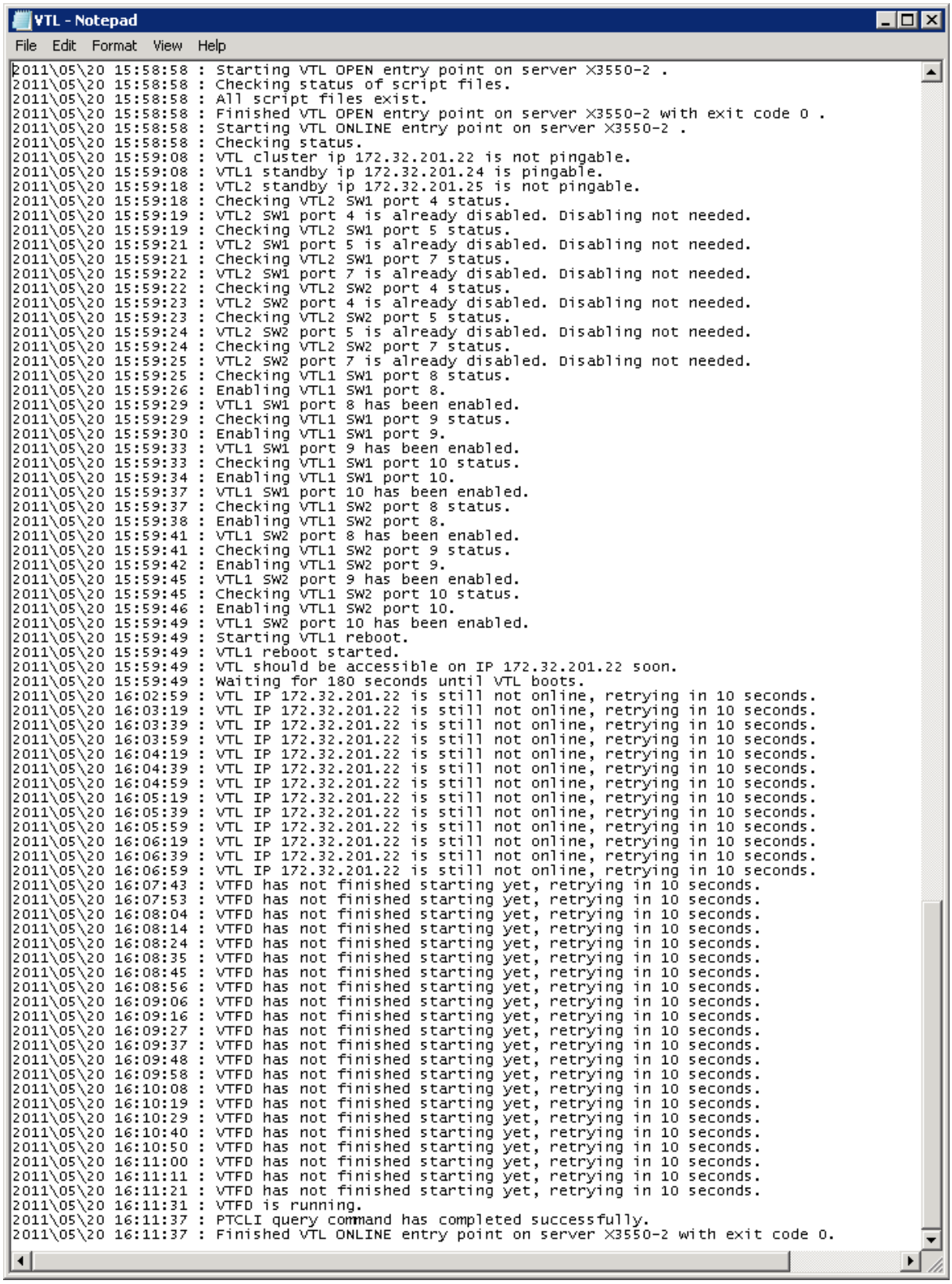
Slika 20. Uspešno zaključen preklop generičnega skriptnega vira.

V dnevnikih generičnega skriptnega vira na sekundarnem in primarnem strežniku lahko vidimo korake ki jih je ta opravil za aktiviranje primarnega vozlišča TS7650G. Na sliki 21 vidimo korake prehoda generičnega skriptnega vira v stanje napake. Generični skriptni vir je 13 sekund po izklopu mrežne vtičnice zaznal, da se trenutno aktivno vozlišče TS7650G ne odziva več, zato je začel s koraki za postavitev tega vozlišča v stanje pripravljenosti. Ker se vozlišče ne odziva, samega vozlišča ne more postaviti v stanje pripravljenosti, zato nadaljuje z izklopom vseh vtičnic na FC stikalih, ki so povezane na to vozlišče TS7650G.



Slika 21. Dnevnik prehajanja generičnega skriptnega vira v stanje napake.

Po končanih korakih na sekundarnem strežniku se generični skriptni vir preseli na primarni strežnik in začne s preklapljanjem primarnega vozlišča TS7650G iz stanja pripravljenosti v stanje delovanja. Na sliki 22 vidimo korake ki jih je naredil generični skriptni vir, oziroma skripte, ki jih je poklical. Kot vidimo iz zapisov v dnevnik je generični skriptni vir najprej preveril stanje ostalih Perl skript. Za tem je preveril stanje obeh vozlišč TS7650G na podlagi tega, kateri IP naslovi so trenutno aktivni. Ko je ugotovil, da je primarno vozlišče TS7650G v stanju pripravljenosti, je nadaljeval z vtičnicami na FC stikalih. Ker so bile vse vtičnice na FC stikalih, ki so priklopljene na sekundarno vozlišče TS7650G, izklopljene, jih ni izklapljal. Vtičnice, ki so priklopljene na primarno vozlišče TS7650G so bile izklopljene, zato jih je vklopil. Na koncu je le še ponovno zagnal primarno vozlišče TS7650G in počakal, da se je postavilo in da je servis *vtfd* končal z izvajanjem.



Slika 22. Dnevnik prehajanja generičnega skriptnega vira v stanje delovanja.

7 SKLEPNE UGOTOVITVE

V sklopu diplomske naloge je bila razvita in testirana rešitev implementacije sistema TS7650G v gruči WSFC. Na ta način smo zagotovili visoko razpoložljivost sistema, saj se v primeru napake, ki povzroči izpad trenutno aktivnega vozlišča TS7650G, avtomatično aktivira drugo vozlišče TS7650G. Vsi koraki, ki jih skripte naredijo pri preklopu, se zapisusujejo v dva dnevnika, kar nam poenostavi reševanje problemov takrat, ko se pojavijo.
Z implementacijo IBM TS7650G sistema smo pokazali kako lahko praktično vsak računalniški sistem implementiramo v WSFC gručo in s tem zagotovimo visoko razpoložljivost. Ker je rešitev večinoma razvita v skriptnem jeziku Perl, jo lahko relativno enostavno prenesemo tudi na Unix in Linux operacijske sisteme.

Rešitev, ki smo jo razvili, bi lahko dodatno izboljšali in jo tako naredili še bolj robustno z naslednjimi izboljšavami:

- Detekcija in odpravljanje možnega IP konflikta: ranljivost naše implementacije sistema TS7650G v gruči WSFC je, da vozlišče TS7650G po ponovnem zagonu uporabi privzet IP naslov. Ker imata obe vozlišči TS7650G enak privzet IP naslov, lahko pride do situacije, ko bi to pripeljalo do IP konflikta.. Do tega bi lahko prišlo naprimer v primeru da bi na trenutno pasivnem vozlišču TS7650G prišlo do začasne prekinitve napajanja na obeh napajalnikih, kar bi prožilo ponovni zagon sistema. Tako situacijo bi na primer lahko rešili z dodajanjem statičnega zapisa v ARP tabelo, s čimer bi bil IP naslov fiksno vezan na MAC naslov. Ko bi v ARP tabelo dodali statični zapis, bi tako lahko vozlišču TS7650G spremenili IP naslov, s čimer bi odpravili IP konflikt.
- Kroženje dnevnikov: naša rešitev vsak pomemben korak, ki ga naredi zapisuje v dva dnevnika, pri čemer pa ne preverja velikosti teh dveh dnevnikov. Dobra praksa pri zapisovanju v dnevnik je, da sistem sam izbriše stare zapise in s tem prepreči, da bi dnevnik zasedel celoten datotečni sistem. V našem primeru se v praksi sicer ne bi smelo zgoditi, da bi kateri od dnevnikov zasedel celoten datotečen sistem, saj pride do zapisovanja v dnevnik samo v primeru preklopa med vozliščema TS7650G, kar pa je relativno redko. Kljub temu možnost, da bi dnevnika zapolnila datotečni sistem, vseeno ni izključena, zato bi bila taka izboljšava sigurno vredna razmisleka.

- Vključiti nadzor tudi nad pasivnim vozliščem TS7650G: dokler ne pride do preklopa med vozliščema TS7650G, naša rešitev ne javi napake v primeru, da bi pasivno vozlišče TS7650G postalo neodzivno, saj periodično preverja samo aktivno vozlišče TS7650G. To bi lahko izboljšali tako, da bi v periodično preverjanje dodali tudi preverjanje pasivnega vozlišča TS7650G. V primeru, da bi zaznali, da pasivno vozlišče TS7650G ni dosegljivo, bi sistem obvestil dežurnega operaterja, ki bi odpravil napako. S tem bi sistem izboljšali do te mere, da bi odkrili napako takoj, ko bi do nje prišlo, s čimer bi se lahko izognili potencialnim problemom pri preklopu med vozliščema TS7650G.

Naša rešitev se trenutno pospešeno testira pri večjem farmacevtskem podjetju in deluje v skladu s pričakovanji z izjemo določenih robnih primerov, kjer deluje nekoliko nenadzorovano, posledično pa preklop traja nekoliko dlje časa. Glede na to, da do teh robnih pogojev v normalnih pogojih ne bi smelo priti, lahko rečemo, da je naša rešitev pripravljena na implementacijo v produkcijsko okolje.

LITERATURA

[1] Alex Osuna, Reimar Pflieger, Lothar Weinert, Xu X Yan, Erwin Zwemmer, IBM System Storage TS7650 and TS7650G with ProtecTIER, 2010

[2] Jon Tate, Fabiano Lucchese, Richard Moore, Introduction to Storage Area Networks, 2006

[3] Jon Tate, Pall Beck, Angelo Bernasconi, Werner Eggli, Implementing the IBM System Storage SAN Volume Controller V5.1, 2010

[4] Werner Vogels, Dan Dumitriu, Ken Birman, Rod Gamache, Mike Massa, Rob Short, John Vert, Joe Barrera, Jim Gray, The Design and Architecture of the Microsoft Cluster Service -- A Practical Approach to High-Availability and Scalability, Munich, Germany, 1998

[5] (2011) ActivePerl. Dostopno na: http://www.activestate.com/activeperl

[6] (2011) Fibre Channel. Dostopno na: http://en.wikipedia.org/wiki/Fibre_Channel

[7] (2011) Microsoft Cluster Server. Dostopno na: http://download.microsoft.com/download/5/B/D/5BD5C253-4259-428B-A3E4-1F9C3D803074/WS08%20R2%20Failover%20Clustering%20White%20PaperTDM.docx

[8] (2011) Perl. Dostopno na: http://en.wikipedia.org/wiki/Perl

[9] (2011) PsTools. Dostopno na: http://technet.microsoft.com/en-us/library/bb896649.aspx

[10] (2011) PuTTY. Dostopno na: http://en.wikipedia.org/wiki/PuTTY