HO
GENT

Faculteit Bedrijf en Organisatie

Een analyse van z/OS Health Checker en hoe deze te optimaliseren

Jonas Braem

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Thomas Pollet
Co-promotor:
Kevin Somers

Instelling: HCL Technologies

Academiejaar: 2019-2020

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Een analyse van z/OS Health Checker en hoe deze te optimaliseren

Jonas Braem

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Thomas Pollet
Co-promotor:
Kevin Somers

Instelling: HCL Technologies

Academiejaar: 2019-2020

Tweede examenperiode

Woord vooraf

Samenvatting

De mainframe is een vitaal onderdeel van verscheidene belangrijke sectoren. Ze worden voornamelijk gebruikt door hun garantie van continue toegankelijkheid. Er is zowel software als hardware speciaal ontworpen voor dit doeleind. z/OS Health checker is een voorbeeld van software dat hiervoor gebruikt wordt. Dit is een preventie tool die problemen probeert op te sporen voordat ze gebeuren. Maar de logging hiervan is niet zo efficiënt en de standaardopstelling van deze tool is ook niet efficiënt voor elke opstelling van en mainframe. Dit is ook het geval bij de opstelling HCL Technologies BVBA het doel van deze proef is die opstelling optimaliseren en een veel efficiëntere manier van logging opstellen. Er wordt eerst gekeken naar structuur binnen de mainframe waar z/OS health checker opereert. Dan naar de opstelling van z/OS Health Checker zelf en een analyse van die opstelling binnen HCL Technologies BVBA. Dan zal er een efficiëntere logging opgezet worden via JCL - jobs met als vervolg een optie deze om te zetten naar een Web interface.

Inhoudsopgave

Lijst van figuren

1. Inleiding

Men zal het niet beseffen maar zonder de Mainframe zouden veel hedendaagse diensten wegvallen. Ook al is de mainframe voor velen iets uit het verleden blijven veel sectoren er op vertrouwen. Bijvoorbeeld de banken, de kans is groot dat als je iets betaalt met je bankkaart dat de transactie verwerkt wordt door een mainframe. Een ander voorbeeld zijn de vliegmaatschappijen, als je eens een kijk zou kunnen nemen naar de computer die gebruikt word op de luchthaven om je in te checken op je vlucht, zal je merken dat dit hoogstwaarschijnlijk een terminal is die aangesloten zit op een Mainframe. Verder wordt de mainframe ook nog gebruikt in andere sectoren zoals:

- Financiële Sector
- Magazijnbeheer
- Verzekeringen
- Ziekenzorg
- Overheid
- ...

Een uitval van een mainframe kan dus kritische diensten laten wegvallen. Bijvoorbeeld de mogelijkheid tot overschrijven via de bank. Daarom is een van de belangrijkste factoren van de mainframe de toegankelijkheid. De mainframe garandeert een uptime van 24 op 24, 7 op 7, 365 op 365 het hele jaar door dus. Hiervoor heeft het al verscheidene technieken waaronder de alles 2 regel. Je zult in een mainframe alles dubbel vinden. Ook de architectuur binnen de mainframe, de Parallel Sysplex(Meer hierover in de hoofdstuk 2) zorgt dat bij een uitval van 1 partitie een andere zijn workload direct overneemt. Dan zijn er binnen die architectuur ook nog verschillende software componenten hiervoor. Een daarvan is degene waar deze proef zich op focust z/OS Health Checker, deze tool werkt preventief. Hij probeert dus problemen op te sporen voor ze kunnen plaatsnemen en

verwittigd de System Administrator hiervan. In hoofdstuk 2 zal de werking hiervan tot in detail worden uitgelegd.

1.1 Probleemstelling

De z/OS Health Checker opstelling van HCL Technologies is al lang niet meer verandert en is ook niet gestandaardiseerd over de verschillende partities binnen de Mainframe. Bij eventuele uitbreiding van het systeem is er dus ook niet direct een standaardopstelling die men kan toepassen. Verder vind er ook een inefficiënte logging plaats. Deze is nu niet echt aanwezig, de bedoeling is dat de verantwoordelijke van elke partitie een log krijgt van alle fouten binnen zijn partitie.

1.2 Onderzoeksvraag

Wees zo concreet mogelijk bij het formuleren van je onderzoeksvraag. Een onderzoeksvraag is trouwens iets waar nog niemand op dit moment een antwoord heeft (voor zover je kan nagaan). Het opzoeken van bestaande informatie (bv. “welke tools bestaan er voor deze toepassing?”) is dus geen onderzoeksvraag. Je kan de onderzoeksvraag verder specifiëren in deelvragen. Bv. als je onderzoek gaat over performantiemetingen, dan

Deze proef zal zich focussen op de z/OS Health checker opstelling van HCL technologies met volgende onderzoeksvraag.

- Is het mogelijk een standaardopstelling te maken voor een z/OS Health Checker omgeving?

1.2.1 Deelonderzoeksvragen

Daarnaast is er in deze proef ook een focus op verdere efficiëntere logging van z/OS Health Checker en of deze kan via een webUI. Daaruit volgen de deelonderzoeksvragen:

- Kan er een efficiënt log-systeem opgezet worden voor de z/OS Health Checker output?
- Is de het mogelijk om de output van z/OS Health Checker vanuit SDSF te loggen op een Web interface?

1.3 Onderzoeksdoelstelling

Het doel van deze proef is om een standaardopstelling te bekomen van z/OS Health Checker binnen de omgeving van HCL Technologies zodat deze bij uitbreiding van het systeem deze onmiddellijk kan implementeren op nieuwe partities.

Verder is het doel om ook een duidelijke logging op te stellen voor elke verantwoordelijke van elke logische partitie van de Mainframe omgeving van HCL, met als eventueel vervolg een mogelijkheid te vinden om die logging te laten gebeuren via een web interface.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de hoofdonderzoeksvraag en de 1ste deelonderzoeksvraag.

In Hoofdstuk 3 wordt onderzocht naar de 2de deelonderzoeksvraag

In Hoofdstuk 4, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2. Stand van zaken

Dit onderzoek zal zich focussen op z/OS Health Checker dit is een tool die draait in een mainframe omgeving maar vooraleer we kunnen beginnen met het bespreken van de oplossingsmethode van de onderzoeksvragen moeten we ons eerst verdiepen in de mainframe zelf en de omgeving waarin z/OS Health Checker draait om zo duidelijk te maken waarom deze tool zijn aanwezigheid belangrijk is en hoe deze werkt. Verder moeten we ook de basis begrijpen van andere tools en systemen binnen de mainframe zoals ISPF, SDS en JES. Om daarna te eindigen met JCL de taal die word gebruikt om de z/OS Health Checker logs op te stellen.

2.1 De Mainframe

De Mainframe speelt een centrale rol in de dagelijkse operaties bij de meeste grote bedrijven. De Ontwikkeling van de mainframe gaat terug tot de jaren ’50. Ook al is er door de jaren heen veel verandert aan de mainframe blijft het het meest stabiele,veilige en compatibele computing platform. Desondanks dat de mainframe een grote aanwezigheid heeft binnen de financiële wereld, blijft deze vrij onzichtbaar voor de grote menigte. Maar eigenlijk zijn we bijna allemaal indirect mainframe gebruikers ook al realiseren we het niet. **Ebbers2011**

De term mainframe kan vandaag het beste beschreven worden als een stijl van operaties, applicaties en besturing systeem faciliteiten. Een definitie hiervan zou zijn: "Een mainframe is hetgeen dat bedrijven gebruiken voor het hosten van commerciële databanken, transactie servers en applicaties de een hogere graad van security en availability nodig hebben dan die in machines van een kleinere schaal. De term mainframe is duidelijk

vervaagd met de jaren daarom wordt de term meestal geassocieerd met een systeem met volgende attributen. **Ebbers2011**

- Compatibiliteit met System z besturingssystemen, applicaties en data.
- Gecentraliseerde controle van resources
- Hardware en besturingssystemen die toegang kunnen delen tot disk drives met ander systemen met automatische locking en bescherming tegen destructief simultaan gebruik van data.
- Hardware en besturingssystemen die continu werken met honderden of duizenden simultane input-output operaties
- Clustering technologieën bevatten zoals de Parallel Sysplex die de mainframe flexibel en schaalbaar maken
- Geoptimaliseerd voor input-output business gerelateerde data processing applicaties

Figuur 2.1: Een paar mainframes, links de IBM z systems z13 en rechts de LinuxOne Rockhopper

2.2 Logische partities en de Parallel Sysplex

Nu men weet wat een mainframe is, is het belangrijk om te kijken naar de architectuur binnen het systeem zelf. Dit is aan de hand van allerlei gekoppelde logische partities die men als geheel de Parallel Sysplex noemt. Dit is ook de omgeving waarin z/OS Health Checker opereert. Daarom belichten we ook deze componenten in dit hoofdstuk.

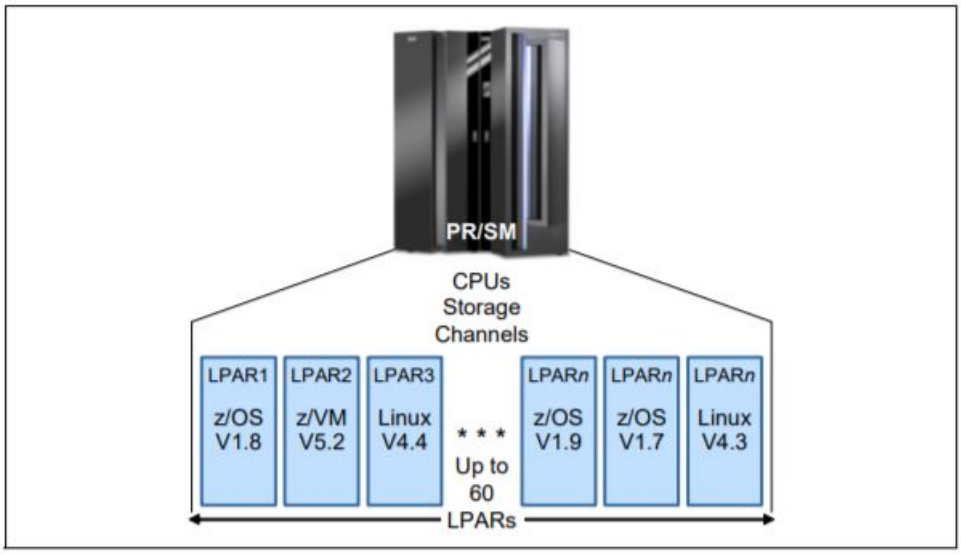
2.2.1 Logische partitie of LPAR

De IBM mainframe kan verdeeld worden in verscheidene logische systemen. Tussen deze systemen kan men volgende resources verdelen:

- Memory
- Processors

- Input-Output devices.

Deze aparte systemen noemt men een logische partitie of LPAR. Al deze LPARs staan onder de controle van een hyperisor. De hypervisor is een software laag voor het beheer van meerdere besturingssystemen. De Verdeling van resources gebeurt door de Processor Resource/Systems Manager(PR/SM). De volledige definitie van een LPAR luidt als volgt: Een subset van de processor hardware dat gedefinieerd is voor het ondersteunen van een besturingssysteem. Meerdere LPARs zijn dus gelijkaardig aan verschillende aparte mainframes. Ze hebben elk hun eigen besturingssysteem en toegewezen hardware. **Ebbers2011**



Figuur 2.2: Visualisatie van logische partities

2.2.2 Parallel Sysplex

De Parallel Sysplex is een een techniek van clusteren. Waarmee men meerdere LPARs groepeert. z/OS Health Checker zal zowel opereren op aparte LPAR als op de gehele Parallel Sysplex door globale checks(meer hierover in sectie 2.8). Daarvoor gaan we ons ook verdiepen in deze clustering techniek.

Sysplex staat voor SYStems comPLEX dit is een of meerdere LPARs met z/OS, samengevoegd als 1 unit die gespecialiseerde hardware en software gebruikt. Het gebruikt unieke messaging services en kan bestandsstructuren delen in de couple facilty(CF) datasets. Een sysplex is een instantie van een computer systeem dat draait op 1 of meerdere fysieke partities waarvan elke een andere release kan draaien van het z/OS besturingssysteem. Een sysplex is wel geïsoleerd tot 1 fysieke mainframe. De Parallel Sysplex anderzijds laat meerdere mainframes zich voordoen als 1 systeem. **Ebbers2011**

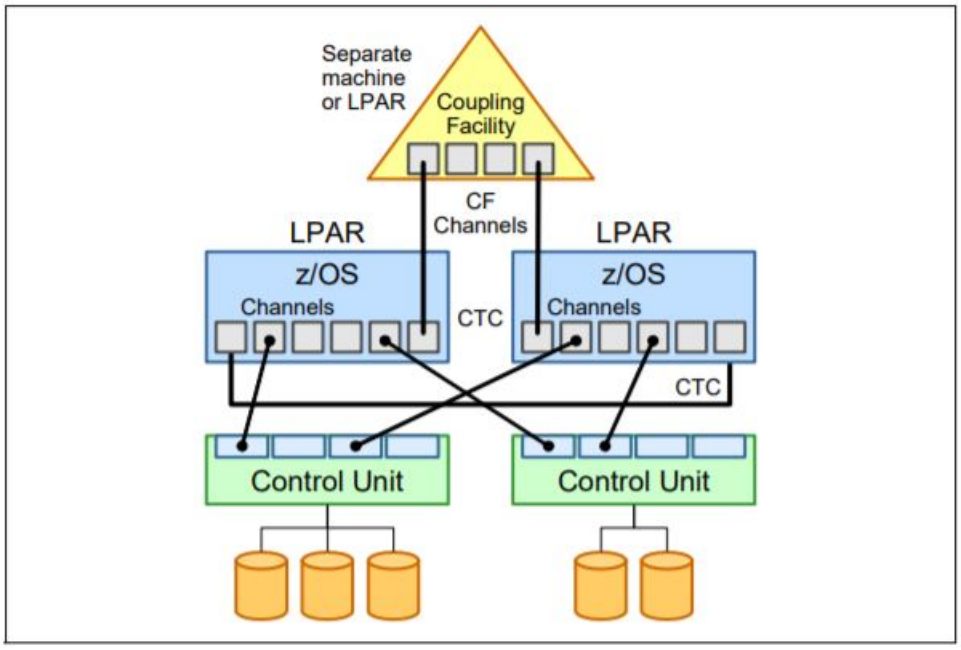
Een Parallel Sysplex is een symmetrische sysplex die gebruik maakt van het delen van data met meerdere systemen. Dit is dus de clustering van meerder mainframes. We bespreken ook enkele protocollen die de Parallel Sysplex gebruikt.

Server Time Protocol

Een belangrijk aspect van de Parallel Sysplex is het synchroniseren van de Time Of Day(TOD) klokken van de meerdere servers. Stel nu meerdere systemen hebben net in dezelfde database data aangepast, maar daarna gebeurt er een uitval. Dan zal men de databank reconstrueren met behulp van alle timestamps van alle aanpassingen. Hiervoor is het belangrijk dat de klokken van elke LPAR gesynchroniseerd zijn om zo de juiste data in de juiste volgorde te reconstrueren. Dit gebeurt vandaag met het Server Time Protocol(STP). **Ebbers2011**

Coupling Facilty

Sommige z/OS applicaties op verschillende LPARs hebben vaak toegang nodig to dezelfde informatie. Hiervoor betrouwt een Parallel Sysplex op een of meerder Coupling Facilities(CF). Een CF maakt het mogelijk om aan data sharing te doen met meerdere systemen. Een CF is ook een LPAR maar een speciale die andere LPARs toelaat data te delen. **Ebbers2011**



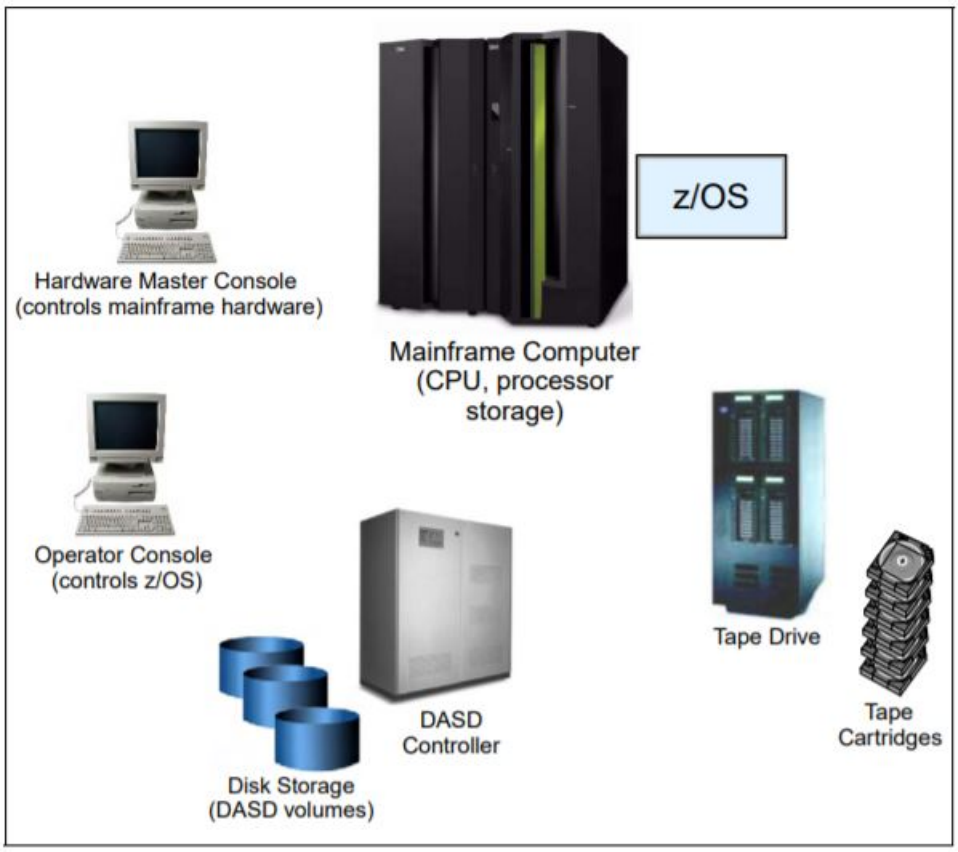
Figuur 2.3: Visualisatie van een parallel sysplex met 2 LPARs en 1 Coupling Facility. Control Units controleren de logica voor bepaalde I/O-apparaten zoals printers of opslagfaciliteiten.

Een goed geconfigureerde parallel sysplex cluster is ontworpen zodat hij een hoge availability aanbied met een minimale downtime. Dit is dus al een van technieken die de mainframe zijn hoge availabilty garanderen. Bijvoorbeeld: als een systeem een uitval zou hebben, zou een ander systeem het direct kunnen opvangen doordat alle data en kritieke applicaties gedeeld worden in de parallel sysplex deze zou de taken van het systeem overnemen en ondertussen zou het uitgevallen systeem zich herstarten. Dit zorgt er uiteindelijk voor dat een laag aantal single points of failure aanwezig is binnen de sysplex. **Ebbers2011**

2.3 z/OS

Ook belangrijk component van elk systeem is natuurlijk het besturingssysteem. In het geval van deze proef is dat z/OS, dit niet het enigste maar wel het meest gebruikte besturingssysteem op de mainframe. Zoals de naam het zelf als zegt draait ook z/OS Health Checker op dit besturingssysteem. Daarom bespreken we ook dit besturingssysteem. Een besturingssysteem is eigenlijk een collectie van programma's die de interne werking van het computer systeem beheren. Een besturingssysteem is ontworpen om er voor te zorgen dat de resources van de computer optimaal gebruikt worden.

z/OS is vandaag een resultaat van tientallen jaren technologische vooruitgang door IBM. Het begon als een besturingssysteem dat maar 1 programma tegelijk kon afhandelen naar een dat vandaag duizenden programma's en gebruikers tegelijk kan afhandelen. Het besturingssysteem wordt uitgevoerd in de processor en bevindt zich ook in de processor storage. Mainframe hardware bestaat uit een aantal processors en aanhangende toestellen zoals DASD(Direct Acces storage devices de mainframe term voor hard disks). Die worden dan allemaal aangestuurd vanuit de consoles gekoppeld aan de mainframe. De DASD's worden gebruikt voor systeem functies of door programma's van gebruikers die uitgevoerd worden door z/OS. **Ebbers2011**



Figuur 2.4: z/OS binnen de hardware omgeving bevindt zich op de mainframe. En wordt aangestuurd door aangesloten consoles/terminals. Om dan de data te bewerken op tape drives of DASD(term voor hard disk binnen de mainframe wereld)

z/OS maakt het ook mogelijk om aan multiprocessing en multiprogramming te doen. Hierdoor maakt het z/OS geschikt voor het uitvoeren van programma's die veel input/output operaties nodig hebben.

Multiprocessing

Dit is het simultaan opereren van meerdere processors die meerdere hardware resources delen zoals memory of externe opslag

Multiprogramming

Multiprogramming laat z/OS toe om duizenden programma's te draaien voor gebruikers die werken aan verschillende projecten waar men zich ook bevindt op de wereld.

2.3.1 Storage gebruik van z/OS

z/OS heeft ook zijn eigen manier voor het gebruik van opslag/storage dit is ook belangrijk om te bespreken, omdat de term 'address space' gebruikt zal worden binnen deze proef. En die term is de techniek die z/OS gebruikt binnen de storage omgeving.

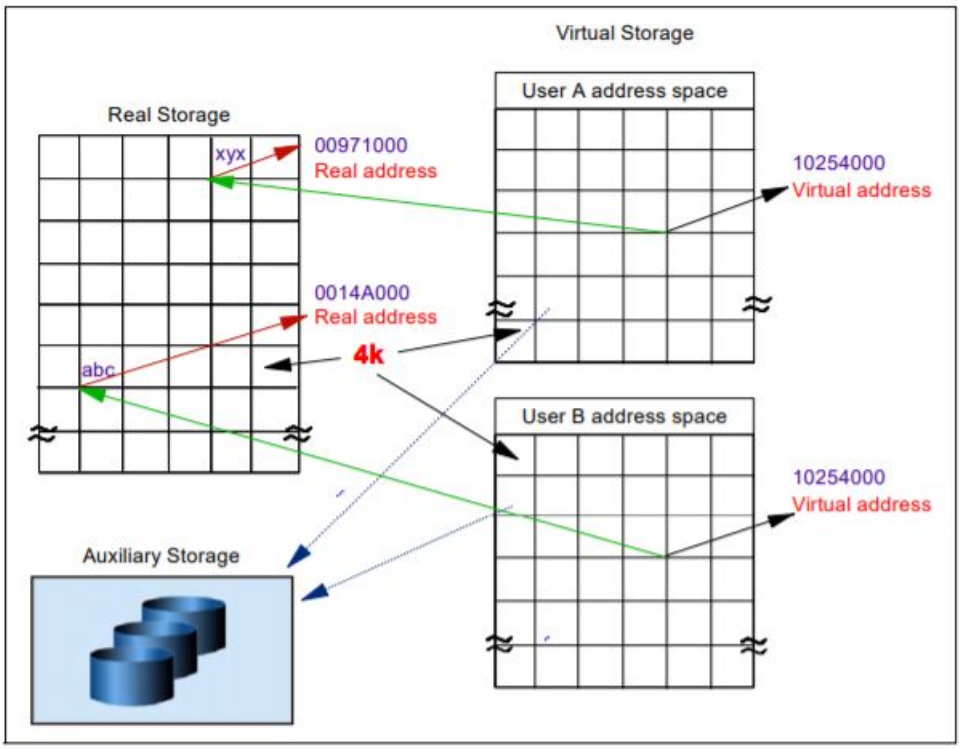
Een mainframe en een gewone computer hebben 2 soorten fysieke opslag:

- Fysieke storage die zich bevindt op de mainframe processor zelf, ook wel 'real storage' genoemd te vergelijken met RAM op je laptop.
- Fysieke storage die zich buiten de mainframe bevindt op een tape of disk drive, dit wordt de auxiliary storage genoemd.

z/OS gebruikt deze 2 soorten storage om een andere soort te vormen namelijk virtual storage. Virtual storage is een combinatie van real en auxiliary storage. Het gebruikt een serie van tabellen en indexen om locaties binnen het real geheugen te associëren met locaties in de auxiliary storage.

Een address space is eigenlijk een range van virtuele adressen dat het besturingssysteem toekent aan een programma of gebruiker. Voor een gebruiker kan dit beschouwt worden als een container waar zijn data in zit. Door deze addres space moet z/OS niet een heel programma naar de real storage laden om het uit te voeren. Daarom zal men het programma in stukken(ook gekend als pages) van de auxiliary storage naar de real storage verplaatsen in de volgorde die nodig is om het prorgamma uit te voeren. Eens dat een page niet meer nodig is kan men het terugschrijven naar de auxiliary storage. Dit laat z/OS toe om meer programma's simultaan uit te voeren.

De fysieke opslag is daarom opgedeeld in verschillende stukken die elke hun eigen adres hebben maar de pages worden opgevraagd met hun virtueel adres. Het proces om een virtueel adres te vertalen in een real adress noemt men Dynamic Address translation(DAT). **Ebbers2011**



Figuur 2.5: Bijna alle programma's gebruiken virtuele adressen als ze refereren naar data in de real storage. Maar als een aangevraagd adres zich niet in de real storage bevindt zal er een onderbreking plaatsvinden en zal men de nodige data uit auxiliary storage naar de real storage laden(ook wel paging genoemd)

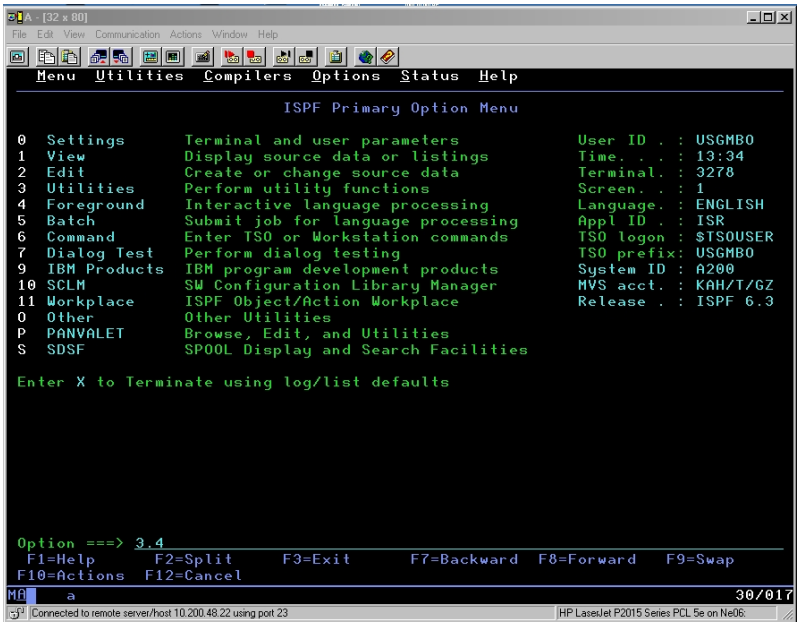
Een andere manier om te denken aan een address space is eigenlijk een soort van kaart voor de programmeur waarmee hij als zijn code en data kan opvragen.

2.4 Interactie met z/OS

Om een systeem te beheren moeten we er natuurlijk ook mee kunnen werken. Voor deze interactie gebruikt men bijvroobeel een TSO/E commando om aan te loggen en dan ISPF, een collectie van menu's en panelen die brede range van functies aanbied.

2.4.1 Time Sharing Option/Extensions en Interactive System Productivity Facility

TSO/E of Time Sharing Option/Extensions laat gebruikers toe om een interactieve sessie te maken met een z/OS systeem. Hierdoor kunnen ze aanloggen op het z/OS systeem en gebruik maken van een command prompt interface. Maar omdat de command prompt niet echt handig is wordt er meestal gebruik gemaakt van de Interactive System Productivity Facility (ISPF). Dit is een collectie van menu's en panelen die een weide range van functies aanbied voor het bewerken van data in z/OS. Zo bied ISPF onder andere een tekst editor aan en functies voor het vinden en lijsten van bestanden. **Parziale2017**



Figuur 2.6: Dit is het hoofdmenu van ISPF vanaf hier kan je de verschillende panelen gebruiken

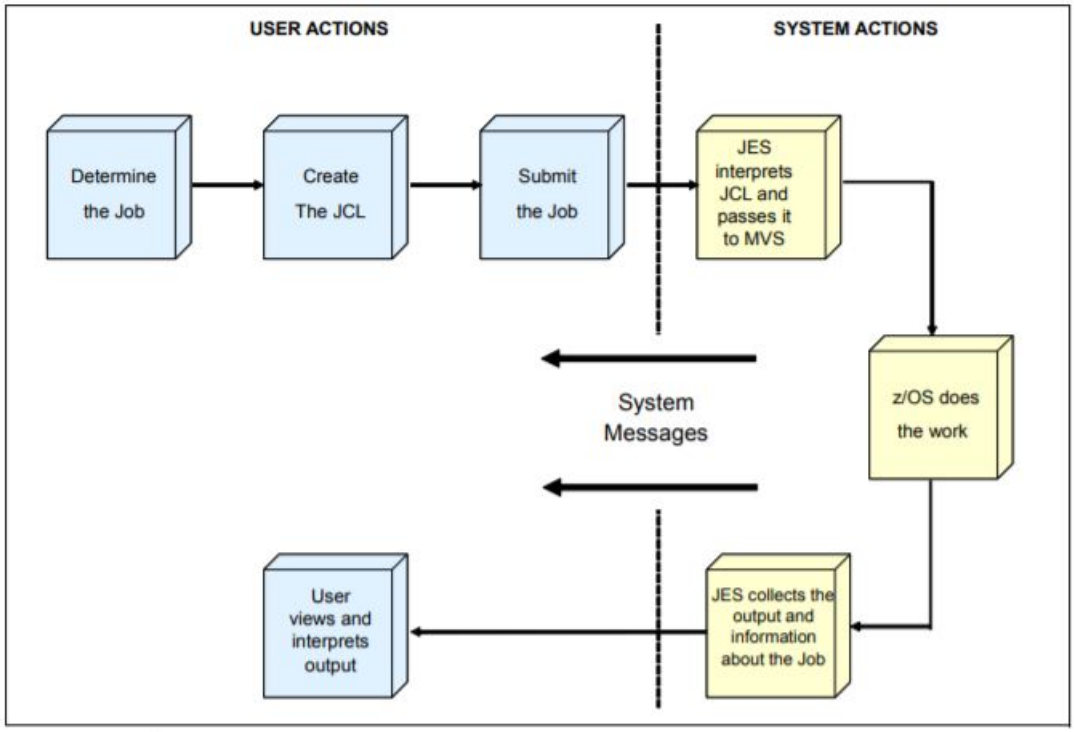
De bestanden binnen z/OS worden worden ook wel 'data sets' genoemd, dit is de term die ook verder gebruikt zal worden als we over bestanden spreken binnen z/OS. Er zijn 2 soorten data sets waarmee gewerkt word in deze proef.

- Sequential data set: dit zijn data sets waarvan de individuele records georganiseerd zijn op hun fysieke volgorde binnen de dataset.
- Partitioned data set(PDS): dit is een data set die verdeeld is in partities ook wel 'members' deze kunnen een programma bevatten of gewoon data. Eigenlijks is een PDS een collectie van sequentiële data sets. Je zou het dus kunnen vergelijken met een folder met bestanden.

2.5 Job Control Language en Job Entry Subsystem

In deze proef worden ook Jobs uitgevoerd voor het opstellen van de logs van z/OS Health Checker. Hiervoor wordt de Job Control Language(JCL) gebruikt. Want om een prorgamma uit te voeren moet het geprocesseerd worden door z/OS.

Vooraleer je het z/OS systeem een programma voor je laat uitvoeren moet je een paar dingen doen. Je moet eerst natuurlijk beschrijven welk programma je wilt uitvoeren maar ook de resources dat deze nodig heeft(bv; eventuele input). Dit doe je met een JCL job. Deze jobs 'submit' je dan naar het Job Entry Subsystem(JES) deze zal de jobs inplannen en hun output verwerken. **Cosimo2018**



Figuur 2.7: De gebruiker definieert, maakt en zal een job submitten. Deze wordt dan verwerkt door het Job Entry Subsystem en zal de output van de job teruggeven als resultaat.

2.6 System Display and Search Facility

We willen natuurlijk de output van onze jobs kunnen zien daarvoor wordt er gebruikt gemaakt van de System Display and Search Facility of SDSF. Dit is niet het enigste dat we kunnen zien in SDSF. Voor z/OS Health Checker zullen we namelijk ook deze tool gebruiken om alle info van het systeem op te vragen. Deze utiliteit kan je bereiken via het hoofdmenu van ISPF met de optie 'S'(zie figuur 2.6)

In SDSF zijn er veel panelen en soorten output die je kan opvragen maar de enige die wij zullen gebruiken is namelijk die van z/OS Health Checker die bereik je binnen SDSF via de 'ck' optie. Verder zal je bij het schrijven van jobs het 'Status of jobs' paneel nodig hebben. Dit paneel toont de output van uitgevoerde jobs, dit paneel bereik je met de 'st' optie.

2.7 De Parmlib

z/OS Health Checker heeft ook enkele datasets in de parmlib die verandert worden in deze proef daarom moeten we ook begrijpen wat die parmlib eigenlijk is. Elk z/OS systeem heeft een PDS dat volzit met members die worden meegegeven door IBM. Deze PDS is SYS1.PARMLIB. De members zijn allemaal systeem en applicatie parameters die het systeem nodig heeft bij het opstarten(Initial Program Load(IPL) in mainframe term). Bij

Figuur 2.8: Hier zie je het hoofdmenu van SDSF en onder andere ook de 2 opties die gebruikt worden doorheen de proef namelijk 'ck' en 'st'

de IPL wordt de parmlib dus gelezen om het systeem op te zetten. Deze PDS wordt later ook nog gelezen door andere componenten en programma's. **Cosimo2018**

Een van deze componenten is namelijk z/OS Health Checker in de parmlib zitten namelijk de members die definiëren welke Checks(meer hierover in sectie) er zullen draaien en welke niet.

2.8 z/OS Health Checker

Na een analyse naar de redenen van verscheidene uitvallen kwam men tot een conclusie dat veel hiervan perfect vermeden konden worden. Vele uitvallen kwamen door slechte configuraties die leiden tot single points of failure. Hierdoor is z/OS Health Checker ontwikkelt door IBM. **Walle2013**

IBM Health Checker voor z/OS is een tool die helpt om potentiële problemen op te sporen in de configuratie van het systeem. Deze problemen zouden een grote impact kunnen hebben op het systeem of zouden zelfs een uitval kunnen veroorzaken. Health Checker kijkt de huidige instellingen van z/OS en de Sysplex na en vergelijkt deze met instellingen die door IBM aangeraden worden. Bij eventuele problemen zal Health Checker een output genereren met gedetailleerde info over het probleem zelf en op welke manier je het probleem het beste oplost. Wel belangrijk is dat Health Checker eerder een preventieve tool is en geen monitoring tool. **Bezzi2010**

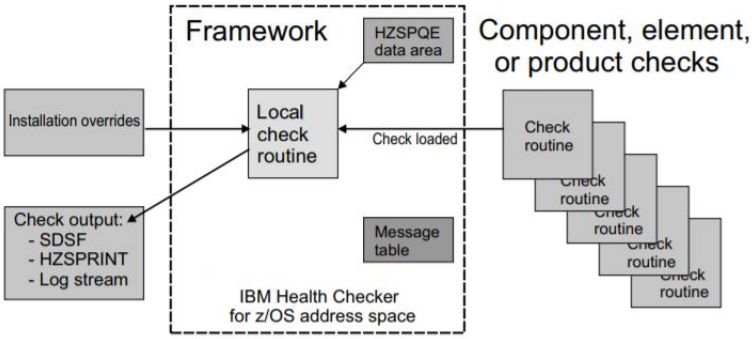
z/OS Health Checker bestaat uit 2 delen

- De checks

- Het framework

2.8.1 Health Checker Framework

Het framework van z/OS Health Checker is de interface die je gebruikt om checks uit te voeren en te beheren. Deze ondersteunt niet enkele checks van IBM maar ook die van software producenten zoals Computer Associates en Compuware. Binnen het framework zit de HZSPQE data, hierin zit alle informatie die een check nodig zou hebben zoals de default parameters. Het framework bevat ook de Message table die houd de data bij van de output van de checks. **IBMCorporation2019**



Figuur 2.9: Het z/OS Health Checker Framework dat het mogelijk maakt om checks uit te voeren en de output hiervan door te geven aan SDSF en andere tools.

2.8.2 De Checks

De Checks zelf zijn programma's die componenten en instellingen evalueren en dan eventuele problemen melden. Deze checks worden voornamelijk aangeboden door IBM zelf maar deze kunnen ook van andere bedrijven zijn. Je kan ook zelf Checks scrhijven met System REXX. Een check vergelijkt de instellingen van 1 bepaald component met een set van instellingen die aangeraden worden door de eigenaar van de check. De aangeraden instellingen zijn geen verplichte instellingen maar eerder een best practice van IBM. Je kan deze ook zelf definiëren of aanpassen. **Bezzi2010**

Een check wordt gedefinieerd met 3 waarden

Check Owner

Elke check heeft zijn eigenaar. De naam van de check is meestal verbonden met het component waarvoor hij draait. In deze naam zit ook bijna altijd een verwijzing naar het bedrijf of product die de check gemaakt heeft. De checks van IBM beginnen dan ook allemaal met IBM. Een check owner is een string van maximum 16 karakters. Een voorbeeld van een check owner is bijvoorbeeld IBMXCF. Deze check is dus geschreven door IBM en zal iets te maken hebben met XCF(Cross-system Coupling Facility). Een check owner heeft meerdere checks zelf. **Bezzi2010**

Check Name

De check zelf heeft natuurlijk ook een naam. En is uniek voor elke check. Deze zal maximum 32 karakters lang zijn. Een voorbeeld van een check naam: "XCF_CDS_MAXSYSTEM".

Check Values

Een check bezit verder ook nog enkele vooraf gedefinieerde waarden

1. Een interval die definieert hoe vaak en wanneer de check uitgevoerd wordt.
2. De ernst(Severity) van een check deze definieert de check output. Hoe grotere het potentiele probleem hoe hoger de severity.

Check Types

Verder zijn er ook 3 types van checks: Local, remote en REXX checks. In deze proef komen enkel de local checks aan bod. De lokale check is geschreven in ASSEMBLER. En draaien binnen de Health Checker address space. Deze worden opgeroepen met parameters die verwijzen naar de HZSPQE data hierin zit alle data die de check routine nodig heeft. Het verschil met een Remote check is dat een remote check niet binnen de addres space van Health Checker draait.

Check Output

De check output is ook zeer belangrijk deze duid namelijk aan of een check al dan niet succesvol was. Er zijn verschillende soorten check messages.

- Information Message: Deze message krijg je wanneer een check succesvol is of wanneer deze niet kan draaien in de huidige omgeving(Bijvoorbeeld wanneer het component dat de check evalueert niet aanwezig is op het systeem).
- Exception Message: Deze message krijg je wanneer de check onsuccesvol was en deze een potentieel probleem heeft gevonden. Dit noemt een exception. Dit bericht bevat de severity van het probleem samen met suggesteis om het probleem op te lossen
- Report: Bij een exception zal er ook een extra report bijgevoegd zijn met extra informatie over het probleem.
- Debug: Sommige checks kan je in debug mode uitvoeren. Dit wordt vooral enkel gebruikt bij het ontwikkelen van een eigen check.

Voor een voorbeeld van check output zie bijlage B.1

Check Status

Verder is er nog 1 speciaal soort check. Namelijk de migration check. Deze check helpen je bij het plannen voor een overschakelen van z/OS versie. Deze checks kan je uitvoeren na een migratie om te kijken of deze succesvol was. Deze checks beginnen altijd met

'ZOSMIG'.

3. Methodologie

Na de stand van zaken die gebaseerd is op de literatuurstudie volgt de volgorde van stappen die ondernomen zijn om deze proef te voltooien.

3.1 z/OS Health Checker standaard setup

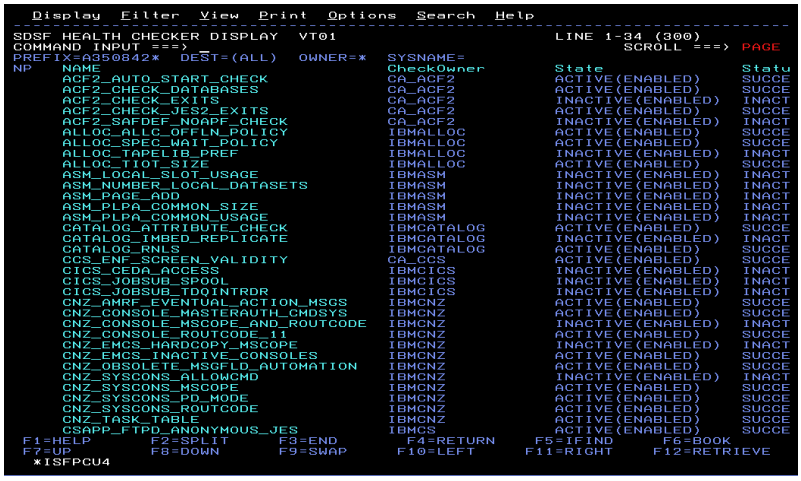
De eerste onderzoeksvraag was of er mogelijkheid was tot een standaardopstelling binnen de z/OS Health Checker omgeving van HCL Technologies. Maar eerst moet er een analyse plaatsvinden op de huidige opstelling van Health Checker. Om deze te optimaliseren

3.1.1 Analyse van huidige z/OS Health Checker Setup

De opstelling die in deze proef geanalyseerd werd bevind zich binnen een parallel sysplex. En deze parallel sysplex werken we met 4 LPARS: VT1, VT2, VT3 en VT4. Elke LPAR heeft verschillende checks. Maar na de 4 LPARS te overlopen was het duidelijk dat de meeste checks op VT1 draaien. Daarom is de analyse gefocust op VT1.

De analyse is gemaakt met de check data uit SDSF deze kan je bereiken door bij het ISPF hoofdmenu volgende optie te geven 's;ck'. Dit is S voor SDSF met als volgende optie CK voor Health Checker deze ziet er zo uit.

Na het overlopen van alle checks op VT1 zijn deze samengevat zoals volgende tabel. Met de naam van de check. De status, deze beschrijft of de check aanstaat of niet. De outcome, deze beschrijft of de check succesvol was of niet. En de reason, dit is de reden waarvoor de check draait.



Figuur 3.1: Dit is het Health Checker paneel binnen SDSF hier kunnen we de output van de laatste keren dat een check is uitgevoerd

Name	Status	Outcome	Reason
XCF_CDS_MAXSYSTEM	ACTIVE(ENABLED)	SUCCES	CDS MAXSYSTEM value across all CDS types should be at least equal to the value in the primary sysplex CDS.

Maar sommige checks hebben GLOBAL als outcome. Dit betekent dat de check niet op de huidige LPAR draait maar op een andere. Global checks draaien voor de gehele sysplex maar moeten maar op 1 LPAR geactiveerd staan.

4. Conclusie

A. Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1 Introductie

Mainframes staan bekend om hun continue toegankelijkheid doorheen hun levensduur. Een van de tools die de toegankelijkheid garandeert is z/OS Health Checker. In dit onderzoek gaat men de werking van z/OS Health checker onderzoeken en volgens met methodes het de mainframe weerhoudt op een eventuele uitval. Daarna wordt de huidige opstelling van deze Tool geanalyseerd binnen de mainframe infrastructuur van HCL. Om deze daarna dan te verbeteren en efficiënter te maken.

A.2 State-of-the-art

De mainframe is en blijft nog altijd een belangrijk systeem binnen onze samenleving. Er wordt vaak gesuggereerd dat de mainframe iets is uit het verleden, maar vele instellingen (bv. Banken) vertrouwen er nog op. Dit is omdat de mainframe garandeert 24 op 24, 365 op 365 operationeel te zijn. Maar voor die garantie heeft het bepaalde tools. Een van deze tools is z/OS Health Checker. Deze tool is geen diagnostische of toezichthoudende tool. Maar ze is eerder een continu lopend preventie tool. Dat potentiele problemen voor de Mainframe probeert te zoeken en te melden aan de systeem administrator. De tool geeft niet enkel het probleem maar ook een aanbevolen actie die de systeem administrator kan nemen.

A.3 Methodologie

Eerst zal er een studie zijn naar de tool op deze te begrijpen en correct te hanteren. Dan pas zullen de opstellingen van z/OS Health Checker op alle systemen van HCL geanalyseerd worden. Met die analyse zal er een vergelijking opgesteld worden om te kunnen onderscheiden op welke systemen de opstelling goed is en op welke deze nog efficiënter kan. Van hieruit wordt er een proof of concept uitgewerkt. Die proberen we dan te implementeren op alle systemen van toepassing.

A.4 Verwachte resultaten

Een duidelijker beeld van hoe een Mainframe zijn toegankelijkheid garandeert met een tool zoals z/OS Health checker. Een andere verwachting is ook een beter begrip van de werking van de z/OS Health Checker tool en hoe deze de mainframe de garantie geeft om 365 op 365 te blijven draaien zonder problemen. Verder zal men ook het belang van een tool zoals deze kunnen begrijpen.

A.5 Verwachte conclusies

Dat z/OS health checker een belangrijk onderdeel is van het mainframe systeem. Dat men na de analyse en vergelijking van de huidige opstelling een betere en efficiëntere heeft ontwikkeld. En dat deze ook geimplemnteerd is en zo de systemen nog betrouwbaarder zijn dan voordien.

B. Bijlagen

B.1 Check Output

Check output bij exception van de XCF_CF_STR_POLICIYSIZE check.

********** TOP OF DATA **********
CHECK(IBMXCF,XCF_CF_STR_POLICYSIZE) SYSPLEX: PLEXVT SYSTEM: VT01
START TIME: 05/12/2020 09:42:52.954735 CHECK DATE: 20090707 CHECK SEVERITY: MEDIUM

IXCH0923I Coupling facility structure policy sizing is summarized by the following report:

An asterisk (*) before a structure name indicates an exception condition. When the qualification is "Alter not allowed", an exception condition is when INITSIZE is specified not equal to SIZE. Otherwise an exception condition is when INITSIZE is less than half of SIZE.

Structure Name INITSIZE Max SIZE Alter Qualification ————————- ——— ——— ——————— *ISGLOCK 20 M 30 M Alter not allowed *MQT0APPL1 256 M 1 G Alter supported *MQT0APPL3 256 M 1 G Alter supported *MQT0APPL4 256 M 1 G No connections defined *MQT0APPL5 256 M 1 G Alter supported

* Medium Severity Exception *

IXCH0255E A CFRM policy structure specification has too large a difference between the INITSIZE and SIZE values.

Explanation: A specification of INITSIZE in the active or pending CFRM policy indicates an initial structure size that is too small for the maximum structure size (as determined by the SIZE specification). Either a structure has an initial size specified as less than half the maximum size, or a structure whose users do not allow structure alter has an initial size specified different than the maximum size.

When allocating the structure initially, whether INITSIZE is specified or not, the system attempts to build all control structures that will be required to support the maximum size of the structure. These control structures are built in the control storage allocation of the structure. For structures whose users do not allow structure alter, the control storage allocated to accommodate larger sizes is wasted. An INITSIZE value substantially smaller than the SIZE value might cause the following:

- It might be impossible to allocate a structure at a size of INITSIZE, because the amount of control storage required to support the SIZE value might actually be larger than INITSIZE.

- If the allocation succeeds, it might result in a structure with a proportionally large amount of its storage allotted to structure controls, leaving too few structure objects to be exploited usefully by the associated application.

System Action: The system continues processing.

Operator Response: N/A

System Programmer Response: IBM suggests that the INITSIZE and SIZE specification for structures be determined by the CfSizer (Coupling Facility Structure Sizer) tool:

http://www.ibm.com/systems/z/cfsizer

Use the CfSizer tool to determine the INITSIZE and SIZE parameters for structures with an exception condition. Update the CFRM policy (or policies) with the new parameters. The new parameters should not have an INITSIZE value for a structure less than half the SIZE value for that structure. If alter is not allowed by users of a structure, INITSIZE should not be specified for that structure. Start an updated policy with the following system command:

SETXCF START,POLICY,TYPE=CFRM,POLNAME=polname

The policy changes will become pending and affect only future structure allocations, not currently allocated structures. REBUILD or REALLOCATE can be used to activate the pending changes for currently allocated structures. For example, a REALLOCATE process can be started with the following system command:

SETXCF START,REALLOCATE

Problem Determination: See IXCH0923I in the message buffer that identifies the coupling facility structures with an exception condition.

The problem may have occurred because a CFRM policy structure SIZE value was adjusted

without also adjusting the INITSIZE value (or vice versa).

The following system command can be used to determine the name of the active or pending CFRM policy:

DISPLAY XCF,POLICY,TYPE=CFRM

Source: Parallel Sysplex (XCF)

Reference Documentation: For more information on planning and activating CFRM policies, see z/OS MVS Setting Up a Sysplex.

For the syntax of the SETXCF START command, see "SETXCF Start Commandïn z/OS MVS System Commands.

Automation: N/A

Check Reason: Too large a difference between INITSIZE and SIZE may waste coupling facility space or prevent structure allocation.

END TIME: 05/12/2020 09:42:53.031335 STATUS: EXCEPTION-MED **********
BOTTOM OF DATA **********