University of Applied Sciences Aachen

Fachbereich Medizintechnik und angewandte Technomathematik

Erstellung eines Parallel-Sysplex in einer virtuellen Umgebung

Diplomarbeit vorgelegt von:

Uli Stormanns Matrikel-Nr: 238597

1. Prüfer Prof. Dr. J. Hoffmann
2. Prüfer Prof. Dr. W. Spruth

Aachen, Wintersemester 2008/09

Zusammenfassung

Zu Zwecken von Forschung und Lehre werden an mindestens zwei Universitäten in Deutschland (Leipzig und Tübingen) z/Series-Computer der Firma IBM betrieben. Auf diesen Rechnern kommen verschiedene Betriebssysteme ebenfalls von IBM zum Einsatz. Dies sind z/VM, z/OS und z/Linux.

Wegen der immer noch herausragenden Bedeutung von z/Series Rechnern insbesondere mit dem Betriebssystem z/OS im Einsatz bei Banken, Versicherungen, öffentlichen Verwaltungen und Industrieunternehmen nicht nur in Deutschland, sondern weltweit wird in dieser Arbeit gezeigt, wie man in einer virtuellen Umgebung eine Kolaboration von mehreren z/OS-Instanzen in Form eines sogenannten Parallel-Sysplex implementieren kann und welche Erkenntnisse sich dabei gewinnen lassen.

Erklärung

Ich versichere, die vorliegende Arbeit selbständig und nur unter Benutzung der angegebenen Hilfsmittel angefertigt zu haben.

Reichenau, den 05. Februar 2009

Danksagung

Prof. Dr. J. Hoffmann von der Fachhochschule-Aachen für die Ermöglichung dieser Arbeit seitens der FH-Aachen
Prof. Dr. W. Spruth von der Universität Leipzig für die Ermöglichung dieser Arbeit seitens der Uni Leipzig
Dieter Begel von der SYSEINS GmbH, für die persöhnliche Betreuung und viele lehrreiche Abende.
Georg Müller, Student der Universität Leipzig, für die Bereitstellung des z/VM's und die gegenseitige Unterstützung.
Meiner Freundin Petra für Ihre Geduld und die Motivation, die Sie mir jeden Tag aufs neue gibt.

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

1 Einleitung

1.1 Aufbau der Arbeit

Die vorliegende Arbeit behandelt das Thema „Erstellung eines Parallel-Sysplex in einer virtuellen Umgebung“ mit Hilfe des Betriebssystem-Clonings.

Zunächst wird in der Einleitung die Geschichte des IBM-Mainframes kurz umrissen. Im zweiten Kapitel werden generelle Aussagen zum Verständnis der Arbeitsweise der z-Series-Architektur gemacht. Dabei wird zwischen der Hardware- und der Software-Seite unterschieden. Eine weitere wichtige Funktion der z-Series-Architektur ist die Virtualisierung. Sie vereint die Bereiche Hardware und Software miteinander. Im letzten Abschnitt des zweiten Kapitels werden die umgesetzten Konzepte und Methoden auf dieser Plattform beschrieben.
Das dritte Kapitel behandelt das Konzept des Parallel-Sysplex. Im ersten Abschnitt dieses Kapitels werden die gebräuchlichsten Formen des Clustering dargestellt. Die folgenden drei Abschnitte beschreiben die zentrale Einheit des Parallel-Sysplexes, die Coupling-Facility. Der letzte Teil des dritten Kapitels behandelt detailliert die Anpassung der Coupling-Facility für ein in dieser Arbeit beschriebenes Szenario mit zwei „Mini-z/OS-Instanzen“ innerhalb eines Parallel-Sysplexes.
Das vierte Kapitel richtet den Focus nochmal auf die Gundlagen, die nötig sind, um einen Masterklon zu erstellen und außerdem mindestens einen Klon vom Masterklon ableiten zu können.
Das fünfte Kapitel befasst sich eingehend mit der Erzeugung des Masterklons und den nötigen Systemanpassungen für einen Betrieb im Sysplex-Modus.
Im sechsten Kapitel wird gezeigt, wie ein Klon von dem zuvor erstellten Masterklon abgeleitet werden kann.
Im abschließenden Fazit werden die Grenzen dieses Konzeptes aufgezeigt und dargestellt, wo bei der Durchführung dieser Arbeit auch die Grenzen der Hardware/Software erreicht wurden.

1.2 Motivation / Zielsetzung

Ziel dieser Arbeit ist die Beschreibung des Aufbaus eines Parallel-Sysplex in einer virtuellen Umgebung. Dabei werden die einzelnen z/OS-System-Instanzen durch ein Verfahren erstellt, das sich „Klonen“ nennt. Zu diesem Zweck wird ein Master-Klon erstellt, der als „Kopiervorlage“ für alle weiteren Instanzen dient. Der Aufbau des Master-Klons und damit auch der anderen Systeme beschränkt sich auf das reine Betriebssystem. Komplexere Subsysteme wie beispielsweise DB2 oder CICS werden als Teil dieser Arbeit nicht installiert.

Nachdem dem Aufbau der z/OS-Systeme werden diese mit Hilfe der Coupling-Facility (CF) zu einem Parallel-Sysplex zusammen geführt. Entsprechend der minimalistischen Ausstattung der z/OS-Instanzen wird die Kopplung nur für die vorhandenen Subsysteme erfolgen, die mit den entsprechenden Einrichtungen für den Multi-System-Betrieb ausgestattet sind.

Auf dem folgenden Bild wird das Zielsystem schematisch dargestellt. Es zeigt, dass diese z/OS-Instanz nur über einige wenige Subsysteme verfügt, soweit diese zum Betrieb absolut notwendig sind. Es fehlen vor allem Subsysteme wie DB2, IMS, CICS, MQSeries, RRS etc..

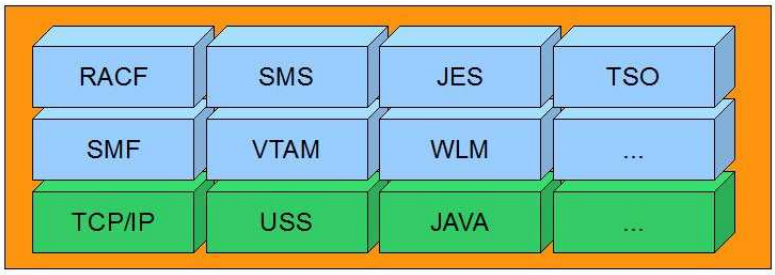

Abbildung 1.1: Schematische Darstellung des Masterklons

Der Betrieb eines Parallel-Sysplex unter diesen Bedingungen hat seine Stärken im Lehrbetrieb, da die Systeme für industrielle oder kommerzielle Anwendungen nicht ausgelegt sind. In kleinen Gruppen, etwa während eines Praktikums, ist es möglich für jeden Studenten ein eigenes System bereit zu stellen, auf dem der Student seine Aufgaben für sich bearbeiten kann. Solche Aufgaben könnten sein:

- „Customizing“ des Systems oder
- Verbinden der Systeme untereinander über die Coupling-Facility oder

- Einrichten und Testen von Funktionen des Workload-Managements in einem Sysplex

1.3 Historisches

Im Jahr 1964 entwickelte die Firma IBM eine Plattform, deren besonderes Merkmal es war, dass sie eine Aufwärts-Kompatibilität mit allen damaligen und den nachfolgenden Modellen dieser Baureihe haben sollte. Die s/360-Rechner waren die ersten Mainframes. Die von den damaligen Entwicklern geplante Kompatibilität ist bis heute einer der Gründe, warum der so oft totgesagte Mainframe immer noch einen so großen Anteil der geschäftsrelevanten Daten verarbeitet, dass dagegen die Anteile der anderen Architekturen fast verschwindend gering sind. Viele Konzepte und Lösungen der Mainfames finden heutzutage bei den PCs kaum bis gar keine Verwendung, während andere gerne adaptiert werden [BSI 1]. Verglichen mit der durchschnittlichen Lebensdauer eines modernen PCs sind die PCs den Mainframes deutlich unterlegen. Die konsequente Abwärtskompatibilität und auch die Ausfallsicherheit, mit der die Maschinen konzipiert wurden, ist wohl die Hauptursache für ihren weit verbreiteten Einsatz. Die Anforderungen an die Availibilty der auf Mainframes laufenden Programme und Services sind enorm. Die Nutzer der Mainframes fordern eine Ausfallsicherheit von 99,999%. Auf ein Jahr umgerechnet entspricht das einer Ausfallzeit von ca. 5 Minuten.
Die s/360-Architektur wurde schrittweise immer weiter entwickelt und an die Bedürfnisse der Kunden, welche vor allem Banken und Versicherungen waren und sind, angepasst. Bei der Entwicklung über die s/370 zu s/390 bis zur heutigen z/Series wurden die Grundelemente beibehalten. Die heutigen Maschinen haben die gleichen Register, den gleichen Maschinencode und die Adressierung läuft auch noch heute nach den gleichen Prinzipien. Die älteren Maschinen sind im Grunde genommen ein Subset der neueren Maschinen.

Zeitleiste

1964 CDC (Computer Data Corp.) verkauft die ersten CDC6600 mit einer Nanosekunde pro Cyklus. IBM kündigt derweil die erste Generation der s/360 Computer an.

1965 IBM verkauft die ersten s/360 Model 40 Computer, die COBOL, FORTRAN und Assembler Programme verarbeiten konnten.

1967 Die ersten s/360 Model 91 werden an die NASA verkauft.

1969 Die ersten s/360 Model 85 werden ausgeliefert. Die s/360 Plattform bietet die Möglichkeit mit verschiedene Betriebssystemen zu arbeiten.

- DOS/360 ⇒ Für kleinere Maschinen. Man kann zwei Echtzeit-Sitzungen und eine Stapelverarbeitungs-Sitzung (Batch) unterhalten.
- OS/360 ⇒ Betriebssystem für Midrange- und Highend-Maschinen.
- TSS/360 ⇒ Betriebssystem für Time-Sharing Multi-User Systeme.

1970 IBM kündigt die s/370 Maschine mit erweitertem Instruktionssatz an. Dieser Typ von Computer ist so populär, dass in der Hochzeit der Nachfrage Kunden bis zu zwei Jahre auf ihre Maschinen warten mussten.
In diesem Jahr wird zum ersten Mal das Sterben der Großrechner prophezeit.

1971 Die ersten s/370 Model 155 und 165 werden ausgeliefert. Dennoch wird weiterhin an den s/360 Maschinen gearbeitet und diese als s/360 Model 195 verkauft.

1973 Einführung der virtuellen Speicherverwaltung für die s/370 Modelle 158 und 168.

1990 Die neuesten Modelle der s/370 Reihe werden nun unter dem Namen s/390 gehandelt. Diese Baureihe waren die erste, die mit Glasfaser-Technik ausgestattet war, die sehr kleinen Schaltkreise erreichten die doppelte Prozessor-Speicherkapazität. Die Basisversion mit luftgekühlten Prozessoren wurde für circa $ 70.500 bis zu $ 3,12 Millionen verkauft. Die mit Wasser gekühlten Maschinen hatten eine Preisspanne von $ 2,45 Millionen bis $ 22,8 Millionen.

1999 IBM stellt die neue Generation der s/390 Serie vor.

2002 Die s/390 G5 und G6 Enterprise Server sind mit bis zu 256 Kanälen, 2 - 8 Kryptographie-Prozessoren und zwischen 8 und 32 GB Hauptspeicher ausgestattet. Es können nahezu beliebig große Plattenspeichersysteme angeschlossen werden. Die Hardware kann von verschiedenen Betriebssystemen gesteuert werden:

- OS/390
- MVS
- VM
- VSE
- TPF

2004 IBM stellt die erste Generation der z/Series vor, den z/890 eServer. Dank der 64-Bit-Technologie und der fortgeschrittenen Virtualisierungsmethoden kann der z/890 mehrere Betriebsysteme zur gleichen Zeit betreiben:

- z/OS
- OS/390
- MVS
- z/VM
- VM/ESA
- VSE/ESA
- TPF
- Linux für zSeries und Linux für s/390

2 Generelle Feststellungen

2.1 Zentrale Hardware-Einrichtungen

IBM teilt die heutige z/Series-Architektur, die im Wesentlichen auf der weiterentwickelten und um die 64-bit-Adressierung ergänzten s/390-Architektur basiert, in folgende Bestandteile auf:

- Hauptspeicher bis zu 512 Gigabyte
- Ein oder mehrere Prozessoren (Central-Processing-Units = CPU)
- Operator-Facilities
- Channel-Subsystem
- I/O-Devices [1]
- zAAP-Processors (integrierte Assist-Processoren für Java- und XML-Applikationen)
- zIIP-Processors (integrierter Informations-Processor zur Entlastung des Primär-Processors)
- optional Integrated Cryptographic Coprocessors
- Fehlerkorrektur-Einrichtungen

2.1.1 Zentrale Einrichtungen/Dienste

Extended Binary-Coded Decimal Interchange-Code

Daten werden auf dem Mainframe im Extended Binary-Coded Decimal Interchange-Code (EBCDIC) verarbeitet und gespeichert. Fast alle anderen Architekturen verwenden den weitaus bekannteren ASCII-Code. Bei der Kommunikation zweier unterschiedlicher Systeme muss also eine Konvertierung durchgeführt werden.
Die IBM-proprietäre EBCDIC-Codierung wurde nicht entwickelt, um sich vom ASCII-Code zu unterscheiden, sondern weil es den ASCII-Code noch nicht gab.

[1] Die I/O-Devices sind über Control-Units mit dem Channel-Sub-System (CSS) verbunden

Register

Besonders schnelle Rechenwerke stellen die Register dar, die für allgemeine und spezielle Aufgaben vorhanden sind. Die verschiedenen Register der z/Series sind alle 64-Bit lang, wobei die Adressierung mit 24- und 32- Bit weiterhin unterstützt wird. Hierzu muss gesagt werden, dass die 32-Bit-Adressierung im Grunde eine 31-Bit-Adressierung ist, da ein Bit als Schalter zwischen 31-Bit- und 24-Bit-Modus benötigt wird. Der Instruction-Operation-Code (IOC) weist die verschiedenen Register zu.

Speicherschutz

Der reale Speicher wird hardwareseitig in vier Kilobyte große Blöcke unterteilt. Pro Block wird ein Speicherschutzschlüssel erzeugt, der bei der Verarbeitung geprüft wird. Durch dieses Verfahren wird ein Überschreiben von fremdem Speicher oder ein Überschreiben durch fremde Prozesse fast unmöglich gemacht.

Operator-Facilities

Die Operator-Facilities sind Werkzeuge, die ausschließlich vom Wartungspersonal oder der Systemadministratoren gebraucht werden. Ähnlich wie beim BIOS eines PCs werden über die Operator-Facilities Systemanpassungen durchgeführt. Die Operator-Facilities sind auf einem Laptop (IBM Thinkpad) eingerichtet, das sich im Gehäuse des Mainframes befindet.

Betriebssystemunterstützung

Betrieben werden kann die z/Series-Hardware mit den entsprechenden Betriebssystemen für diese Plattform. Dies sind

- z/OS
- z/VM
- z/VSE und
- z/Linux

Entsprechend dem Konzept der Entwickler von 1964, dass die aktuelle Mainframe-Plattform abwärtskompatibel zu allen Vorläufern sein soll, kann die aktuelle z/Series sowohl mit 24-Bit oder 31-Bit oder auch, und das ist der heutige Standard, mit einer 64-Bit-Adressierung arbeiten. Bei den modernen Mainframes fällt die 32 Gigabyte Grenze, die auch „The Wall“

genannt wird, weg. Diese Grenze stammt aus Zeiten der Unterscheidung zwischen 31-Bit- und 24-Bit-Adressierung [BSI 2]. Im Rahmen der 64-Bit-Adressierung entfällt auch die Auslagerung von Seiten aus dem Hauptspeicher in den Expanded-Storage, was wiederum einen höheren Durchsatz zur Folge hat.

2.2 Eingabe / Ausgabe - Channel-Sub-System (CSS)

Das Lesen von und Schreiben auf externen Geräten wie Festplatten, Magnetbänder oder Netzwerk-Geräte werden über externe Verbindungen realisiert, die zum Channel-Sub-System (CSS) zusammengefasst werden. Das CSS ist eine eigenständige Einheit bestehend aus Software und Hardware. Es ist als Verteiler zwischen den I/O-Devices zu den Prozessoren zu betrachten. Das CSS besteht aus Kanälen (Channels und Pathes), die wiederum in Unterkanäle (Sub-Channels) aufgeteilt werden. In den Sub-Channels laufen die sog. Channel-Programs. Mit der z/990 wurde das CSS zum Logical-Channel-Sub-System erweitert, welches bis zu 15 CPUs unterstützt.
Auf der Hardwareseite ist das CSS durch einen oder mehr System-Assist-Prozessoren (SAP) implementiert. Die SAPs sind dedizierte Prozessoren, die sich in ihrer Architektur nicht von den Hauptprozessoren unterscheiden. Lediglich der auf ihnen ausgeführte Code ist limitiert.

Über die seriellen Anschlüsse, genannt ESCON (Enterprise System CONnection), werden externe Verbindungen über die ESCON-Directors (das sind Hochleistungs-Switches) in das CSS integriert. Das Multiple-Image-Facility (MIF) erlaubt den Zugriff auf Daten über die Grenzen einer logischen Partition hinaus. Eine Weiterentwicklung der ESCON-Technik, basierend auf Glasfasertechnik, sind die FICON-Anschlüsse(FIbre CONnection), die bis zu 100 MegaBit Bandbreite bereitstellen und gleich den ESCON-Anschlüssen über Directors ins CSS integriert werden.

2.2.1 Telekommunikations-Einrichtungen

Um mit einem Ethernetgerät kommunizieren zu können, wurden OSA-Express-Adapter (Open Systems Adapter) implementiert. Daneben gibt es weitere Telekommunikations-Einrichtungen, die via CSS an den Rechnerkomplex angeschlossen werden oder darin integriert sind [sABC1].
Mit den schnellen Channel-To-Channel-Verbindungen (CTC) wird die Kommunikation zwischen einzelnen Systemen, auf denen unterschiedliche Betriebssystem laufen können, realisiert. CTC-Verbindungen werden auch von Subsystemen und Anwendungen genutzt

wie beispielsweise JES (Job-Entry-Subsystem) und Virtual-Telecommunications-Access-Method (VTAM).

2.2.2 HMC- & MVS-Konsole

Die Host-Managment-Console (HMC) wird benötigt, um grundlegende Einstellungen an der Hardware vorzunehmen. Sie ist für die komplette Maschine, also alle Logical Partitions (LPARs) zuständig. Nicht zuletzt deswegen ist der Zugriff auf die HMC äußerst streng zu reglementieren, denn über die HMC kann unter anderem ein Initial-Program-Load (IPL) ausgelöst werden, was bei einem PC einem Reboot gleichkommt
Die System-Konsolen oder auch MVS-Konsolen dienen zur Steuerung des Betriebssystems. Es ist möglich, mehr als eine MVS-Konsole zu unterhalten, jedoch gibt es nur eine Master-MVS-Konsole und der Zugriff auf diese sollte ähnlich gehandhabt werden wie der Zugriff auf die HMC-Konsole. Man kann MVS-Konsolen entsprechend den Bedürfnissen der Anwender anpassen. So ist es möglich, alle Nachrichten aus dem Tape-Pool auf eine Konsole und alle Nachrichten aus dem JES auf eine andere Konsole umzuleiten.

2.2.3 Externe Daten auf Festplatten und Bändern/Kassetten

Die Speichermedien Festplatten und Bänder/Kassetten spielen natürlich für performante Datenverarbeitung eine besondere Rolle.

Festplatten

Wesentlicher Bestandteil der Peripherie der z/Series Plattform sind die Festplatten. Um die Ausfallsicherheit zu erhöhen, werden nur ausgereifte Plattenmodelle eingesetzt. Die heute meist von Nicht-IBM-Anbietern gelieferten Platten-Systeme verhalten sich kompatibel zu den gängigen IBM-Platten-Modellen vom Typ 3390 (3390 Model 3 hat eine Kapazität von ca. 2,7 GB). Die Volumes werden über Steuereinheiten mit dem z/Series-Server verbunden.

Tapes

Die z/Series-Server und Betriebssysteme unterstützen verschiedene Arten von Magnetbändern, von einzelnen Bandlaufwerken, über Bandroboter, die die Bänder selbstständig verwalten und zur Verfügug stellen, bis hin zu virtuellen Bandsystemen, die die Daten im ersten Schritt auf integrierten Festplatten puffern, um diese im zweiten Schritt sehr performant auf die Bänder zu schreiben. Moderne Band-Systeme müssten korrekterweise Kassetten-Systeme genennt werden [BSI 1].

2.3 Licensed Internal Code (LIC)

Die Verbindung zwischen Hardware und Software wird durch den 'Licensed Internal Code (LIC) hergestellt. Zu vergleichen ist der LIC am ehesten mit dem BIOS eines normalen PCs, aber eine wirkliche Entsprechung im PC-Bereich gibt es für den LIC nicht. Der LIC wird durch einen Vorgang, der Initial-Mircocode-Load (IML) genannt wird, in einen Bereich des Speichers geladen, auf den nur der IML-Prozess schreibend zugreifen kann. Da der LIC sich auf die komplette Maschine bezieht, gilt dies auch für den IML-Prozess. Ein Vorgang der ein IML auslöst, wäre beispielsweise ein Power-On/Reset, also das Einschalten der Hardware am Gehäuse. IML initialisiert somit alle LPARs und damit auch die Betriebssysteme [BSI 1].

2.4 Leipzig

Die an der Universität Leipzig vorhandene Hardware stellt sich so dar:

Abbildung 2.1: s/390 Server derUniversität Leipzig

2.4.1 Übersicht

- S/390 Multiprise 3000 Enterprise Server Model 7060-H70 (2-Way Dual Processor)
- 4 Gigabyte Hauptspeicher
- 256 kByte Prozessor Cache
- SAP (System Assist Processor)
- 14 ESCON-Channels mit jeweils 17 MB/sec
- 3 Ethernet Adapter je 100 Mbit/s
- Enterprise Storage Subsystem Tetragon 2100
- 64 emulierte 3390 Model 3 Platten (Volumes),
  effektiv also ca 172 GB Festplattenspeicher
- Stromversorgung: 380V Kraftstrom-Anschluss

[HostL]

2.5 Software-Betrieb auf z/Series-Rechnern

2.5.1 Das z/OS-Betriebssystem

z/OS ist ein 64-Bit-Betriebssystem für die z/Series-Rechner von IBM, den sogenannten Großrechnern oder Mainframes. Dabei steht 'z' für Zero-Down-Time, was die hohe Fehlertoleranz verdeutlichen soll. Es entstand im Jahre 2001 als Nachfolger des Betriebssystems OS/390. Dieses wurde in 1995 kreiert als Weiterenwicklung von MVS (Multiple Virtual Storage). OS/390 brachte die Erweiterung um die UNIX-System-Services (USS), einer Unix-Komponente. Die Grundbestandteile von MVS sind jedoch in ihrer Struktur immer noch vorhanden, so dass das MVS-Betriebssystem auch heute noch die Basis von z/OS bildet. MVS entstand im Jahre 1974 über mehrere Entwicklungsstufen hinweg aus dem Betriebssystem OS/360 und wurde seitdem ständig weiterentwickelt. Nachfolgend die Grundstruktur des z/OS Betriebssystems:

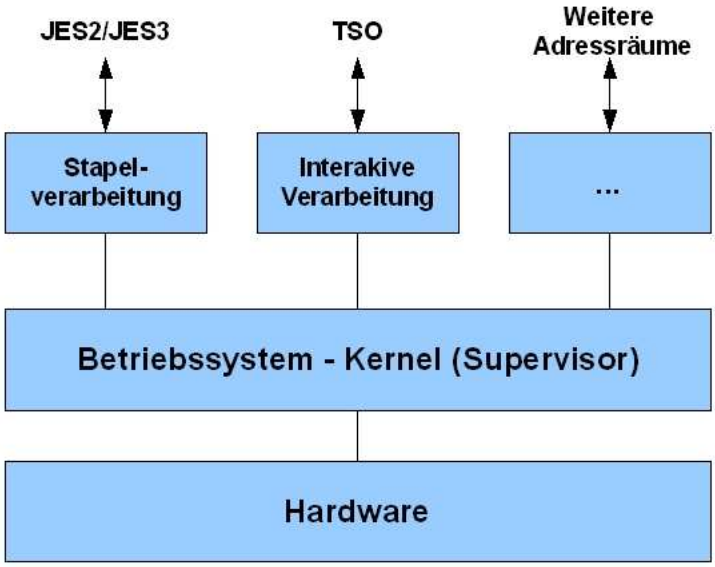

Abbildung 2.2: Grundstruktur des z/OS-Betriebssystems

MVS (Multiple Virtual Storage)

MVS wird es eingesetzt, um große Datenmengen schnell und zuverlässig zu verarbeiten. Es unterscheidet sich in seiner Leistungsfähigkeit von der eines Supercomputers, wie zum Beispiel dem „Blue Gene/L Modell JUBL“ im Forschungs-Zentrum Jülich. Diese können mit höchster Geschwindigkeit komplizierteste Berechnungen durchführen, wobei dann allerdings die I/O-Performance (Ein-/Ausgabe-Leistung) zu kurz kommt. Das Lesen und

Schreiben von riesigen Datenmengen und das Ausführen relativ einfacher Berechnungen mit diesen Daten ist die Stärke von Mainframes mit dem Betriebssystem MVS.
Zum Beispiel würde ein großes Unternehmen für die Gehaltsabrechnung seiner Angestellten oder eine Warenhauskette zur kaufmännischen Verwaltung MVS einsetzen. Da es sich um ein Mehrbenutzer-Betriebssystem handelt, können unter MVS viele Benutzer gleichzeitig mit denselben oder auch unterschiedlichen Programmen und Daten arbeiten.

PC-Benutzer nennen Mainframes oft ein wenig verächtlich Dinosaurier. Verglichen mit MS-Windows- oder Apple-Betriebssystemen erscheint die Oberfläche von MVS primitiv. MVS hatte schon bei seiner Einführung 1974 viele Funktionen, die erst nach und nach in die PC-Betriebssysteme integriert wurden oder werden. Zum Beispiel beinhaltet MVS Dienste zur Fehlerbehebung bei defekten Datenträgern oder sogar bei defekten Prozessoren in einem Mehrprozessorsystem. Die Sicherheit der Daten eines Benutzers ist integraler Bestandteil des MVS-Betriebssystems.

2.5.2 Stapelverarbeitung

Der Name MVS (Multiple Virtual Storage) hat mit der eingesetzten Technik - der Speicherverwaltung - zu tun. Ein Benutzer kann nämlich mit großen Speichermengen arbeiten, so dass sich MVS hervorragend für die Stapelverarbeitung und auch für interaktive Anwendungen eignet.
Ein Stapelverarbeitungs-Job ist das genaue Gegenteil von interaktiven Benutzer. Anstatt interaktiv ein Programm zu starten, Eingaben zu tätigen und auf das Ergebnis zu warten, legt man die benötigten Bestandteile des Jobs schon zu Beginn fest. Man gibt dem System vor, welches Programm auszuführen ist, welche Daten zu verwenden sind und was mit den Ergebnissen zu geschehen hat. Dabei läuft ein Stapelverarbeitungsjob im Hintergrund ab, so dass man während seiner Ausführung andere Tätigkeiten durchführen kann.
Bei der Stapelverarbeitung sind die beiden Subsysteme JES2 oder JES3 von Bedeutung. Dabei handelt es sich um zwei Versionen des Job-Entry-Subsystems, dem Teil des MVS-Betriebssystems, der den Ablauf der Jobs steuert. Die Anweisungen für das JES werden in der Job Control Language (JCL - Job-Steuer-Sprache) geschrieben.
Damit wird festgelegt, wie MVS mit einem Stapelverarbeitungsjob umzugehen hat. Mit JCL legt man die Größe des zu nutzenden Speichers, die Priorität, das Ausgabeziel, die Eingabedateien, die Drucker, die maximale Prozessorzeit und den zu verwendenden Plattenspeicher des Jobs fest. Auch in dieser Arbeit wird JCL für zahlreiche Aufgaben eingesetzt.

2.5.3 Interaktive Schnittstellen

MVS bietet einige Möglichkeiten zur Interaktion - darunter TSO (Time Sharing Option), ISPF (Interactive System Productivity Facility) und weitere Subsysteme.

TSO

TSO ist der Teil von MVS, mit dem das System interaktiv verwendet werden kann. Man kann Befehle eingeben und sieht das Ergebnis auf dem Bildschirm. TSO verfügt außerdem über Einrichtungen zum Starten und Kontrollieren von Stapelverarbeitungsjobs. TSO wurde 1969 zu einer Zeit eingeführt, als die interaktive Verarbeitung eine völlig neue Funktion war, die einen Großteil des Speichers beanspruchte. Daher kommt auch das ‘O‘ im Namen: Am Anfang war TSO eine optionale Komponente, heute ist es Teil jedes MVS-Betriebssystems.

ISPF

ISPF ist eine Anwendung unter TSO. Es ermöglicht das Ausführen von grundlegenden Funktionen über Menüs oder über das Ausfüllen von Formularen oder Panels. Mit seiner eleganteren Oberfläche kann ISPF viele Dinge vereinfachen.
[gSchl]

2.6 Virtuelle Maschinen

Die Emulation ist ein Mechanismus, der es erlaubt, eine Hardware-Architektur in einer anderen Hardware-Umgebung abzubilden. Die abgebildete Architektur nennt man „Gastsystem“ das abbildende (gastgebende) System „Host-System“. Das Host-System ist für die Zuordnung von Systemressourcen wie CPU’s, CPU-Zeit, Hauptspeicher und Ein/-Ausgabegeräte zuständig. Durch entsprechende Software ist es beispielsweise möglich, auf einem „MAC OS-X“-Host-System ein „Microsoft Windows-XP“-Betriebssystem als Gastsystem zu emulieren. Es liegt nahe, dass dies erhebliche Performanceeinbußen mit sich bringt, da jeder Maschinenbefehl des Windows in einer OS-X Umgebung interpretiert werden muss.
Eine Sonderform der Emulatoren ist die Emulation der tatsächlich zu Grunde liegenden Hardware. Diese Form wird „virtuelle“ Maschine genannt. Ein Hypervisor wie z.B. das CP von z/VM (CP=Control Program) oder der Virtual-Machine-Monitor (VMM) ist ein Emulator, welcher besonders effektiv die eigene Systemarchitektur emuliert [Parti]. Die

Emulation schließt die Zentraleinheit mit ihrer Betriebssystem-Konsole (Administrator-Console), den Hauptspeicher sowie die E/A-Einheiten ein. Durch die Emulation der eigenen Hardware ist es möglich, ohne großen Verlust von Performance mehrere Instanzen von einem für diese Architektur geeigneten Betriebssystem oder mehrere verschiedene Systeme gleichberechtigt nebeneinander laufen zu lassen. Bezogen auf die Mainframe-Umgebung heißt das, dass es möglich ist, mehrere z/OS-, OS/390-, VSE-, zLinux- oder VM-Instanzen oder weiter verschachtelte virtuelle Umgebungen zu betreiben [Parti].
IBM bietet mit z/VM als Betriebssystem und/oder PR/SM (Processor Resource/System Manager) als Hypervisor eine Produktkombination an, um die Hardware teilweise oder vollständig zu virtualisieren.

2.6.1 PR/SM (Processor Resource/System Manager)

Der Processor-Resource/System-Manager ist eine Funktion des LIC (siehe oben), die für die unterste Ebene der Virtualisierung zuständig ist. Mit dem PR/SM lässt sich die Hardware in bis zu 30 einzelne logische Partitionen (LPARs) aufteilen. Auf jeder Partition läuft ein von den anderen Partitionen unabhängiges Betriebssystem. Dabei spielt es keine Rolle, welches Betriebssystem auf der einzelnen LPAR läuft. Möglich sind alle in der Zeitleiste (1.3 auf Seite 9) aufgezählten Varianten. Zulässig sind auch Kombinationen, die eine weitere Virtualisierung erlauben, Beispielsweise ein z/VM als Gast eines z/VM's als Gast eines z/VM's ...
Kontrolliert werden die gemeinsamen Ressourcen durch PR/SM.

2.7 Virtualisierung in Leipzig

Die Umgebung für den Parallel-Sysplex auf dem Host in Leipzig ist im Laufe der Jahre gewachsen. Das System ist durch den PR/SM-Hypervisor mit drei logische Partitionen konfiguriert, in denen drei verschiedene Betriebssysteme unterhalten werden (Stand Oktober 2008):

- z/OS V. 1.5
- OS/390 V. 2.7
- z/VM
  - z/Linux
  - z/OS
  - z/VM

Unter dem ersten z/VM werden zLinux, eine weitere Kopie von z/OS und ein weiteres z/VM betrieben. Unter dem zweiten z/VM laufen wiederum verschiedene Instanzen von z/OS, unter anderem auch die für den virtuellen Sysplex relevanten Systeme SYS1Z01 und SYS1Z02 (Abb: 2.3 auf Seite 22).

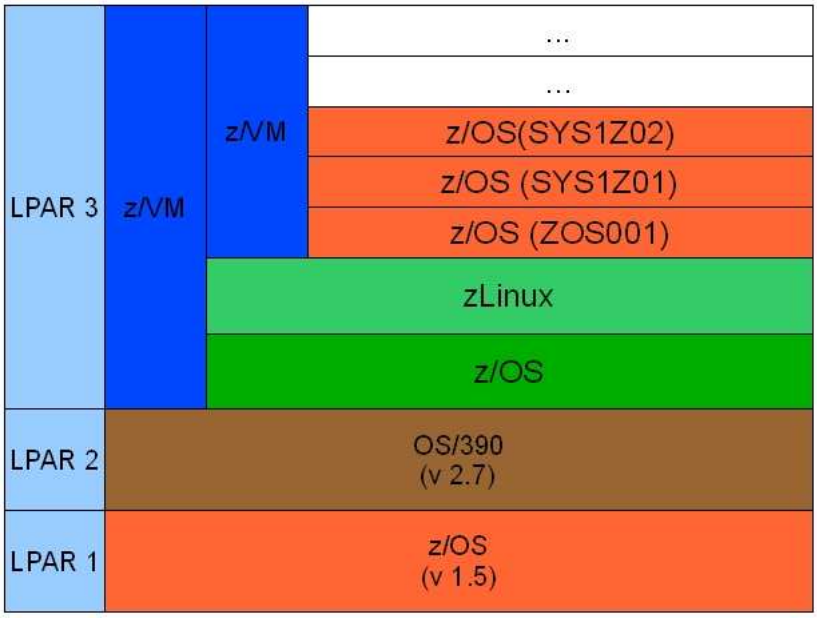

Abbildung 2.3: Virtualisierung in Leipzig

Dieser Umstand wird später noch eine gewichtige Rolle spielen.

3 Parallel-Sysplex

Mit Einführung der ESA-Architektur ab etwa 1990 wurde es möglich, mehrere z/OS-Systeme mit Hilfe einer sogenannten Coupling-Facility zu einem Parallel-Sysplex zusammen zu schliessen. Damit eröffneten sich neue Möglichkeiten zur Skalierung der Systeme unter den Aspekten von steigender Arbeitslast und größerer Ausfallsicherheit. Die beteiligten Systeme konnten auf der selben Hardware und/oder auf einem physikalisch anderen Rechnerkomplex ggf. geografisch voneinander getrennt betrieben werden. Einen solchen Rechnerkomplex nannte man auch Cluster (Deutsch: Haufen).
Die Zielsetzung beim Clustering von mehreren Systemen auf Software- oder Hardware-Ebene ist mehrschichtig und kann in drei wesentlichen Aspekten subsummiert werden:

- **High-Availibility-Cluster (HAC)**
  Um eine möglichst hohe Verfügbarkeit zu gewährleisten, sind Redundanzen hardware- und software-seitig gewünscht. Die zusätzlichen Komponenten können die Arbeitslast eines ausfallenden Systems oder Subsystems fast unterbrechungsfrei übernehmen. Die Verfügbarkeit nimmt zu.

- **Load-Balancing-Cluster (LBC)**
  Das Load-Balancing ist ein Prinzip, das Anwendung findet für besonders rechenintensive Probleme, jedoch mit einer vergleichsweise geringen Komplexität. Als Beispiel für LB-Cluster gilt das SETI-Projekt (SETI=Search for Extraterrestrial Intelligence). Das SETI-Projekt verteilt die Auswertung der Daten auf einen Cluster, der aus vielen tausend privaten PCs besteht. Privatpersonen stellen aus verschiedenen Gründen ihre Rechner dafür zur Verfügung.

  Im Parallel-Sysplex wird dies in ähnlicher Form realisiert. Der Work-Load-Manager verteilt nach entsprechend vordefinierten Algorithmen die Arbeitslast so auf die vorhandenen Komponenten, dass die geforderten Ziele (z.B. tagsüber kurze Antwortzeiten, nachts hoher Stapelverarbeitungs-Durchsatz) erreicht werden.

- **High Performance Computing Cluster (HPCC)**
  Das Prinzip des High Performance Computing Clustering wir zur Berechnung von

komplexen Modellen verwendet. Mit diesem Verfahren werden Wettermodelle oder bewegte Mehr-Körper-Simulationen berechnet. Als Beispiel für einen solchen Cluster ist der IBM-Blue-Gene Cluster JUBL im Forschungszentrum Jülich zu nennen.

In dieser Arbeit ist im Zusammenhang von Clustern eine Kombination von HA- und LB-Clustern gemeint. Im folgenden Abschnitt wird beschrieben, wie der Zusammenschluss der einzelnen Systeme konkret funktioniert bzw. welche Alternativen es bezüglich des Koppelns gibt.

3.1 Kopplung

Es gibt verschiedene Typen der Kopplung bei Clustern. Die Unterscheidungskriterien sind dabei die Zuordnung des externen Speichers und die Art der Prozessor-Kopplung. Betrachtet man die Kopplung von Prozessor und Speicher, kann man zwischen der engen Kopplung (tightly coupled) (Abb.: 3.1), der losen Kopplung (loosely coupled) (Abb.: 3.2) und der nahen Kopplung (closely coupled) (Abb.: 3.3) unterscheiden.

3.1.1 Enge Kopplung

Bei der engen Kopplung (Abb.: 3.1) greifen die verschiedenen Systeme gleichberechtigt über einen gemeinsamen Bus auf einen gemeinsamen Speicher zu, in dem sich eine einzige Instanz des Betriebssystems befindet. Diese Architektur wird auch als Symmetrischer-Multi-Prozessor (SMP) bezeichnet.

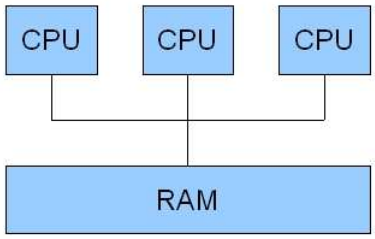

Abbildung 3.1: Enge Kopplung

3.1.2 Lose Kopplung

Die Architektur der losen Kopplung (Abb.: 3.2) zeichnet sich dadurch aus, dass jeder CPU ein privater Speicherbereich zugeordnet ist, in dem sich jeweils eine eigene Instanz des Betriebssystems befindet.

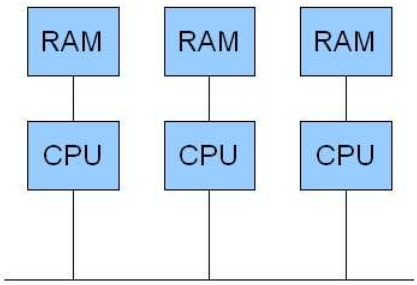

Abbildung 3.2: Lose Kopplung

3.1.3 Nahe Kopplung

Werden die enge und die lose Kopplung miteinander kombiniert, kommt man zu einem Aufbau, der nahe Kopplung genannt wird. Dabei wird jedem Prozessor sowohl ein privater Speicherbereich zugeordnet als auch ein globaler Hauptspeicher bereitgestellt, auf den die Systeme gleichberechtigt zugreifen können. Der globale Speicher wird wiederum von einem eigenem System verwaltet. Der Zugriff auf den globalen Speicher und die dort hinterlegten Daten erfolgt sehr schnell, da dies zum Teil durch eigene Maschinen-Befehle, die aufwändige nachrichtenbasierte Kommunikationsvorgänge überflüssig machen, erledigt wird. Über ein Hochgeschwindigkeitsnetzwerk mit einem Durchsatz von ca. 10 GB/s bis zu 100 GB/s werden die einzelnen Systeme miteinander verbunden. Dies wird in der Regel durch Switches realisiert, die auch parallel geschaltet werden können.

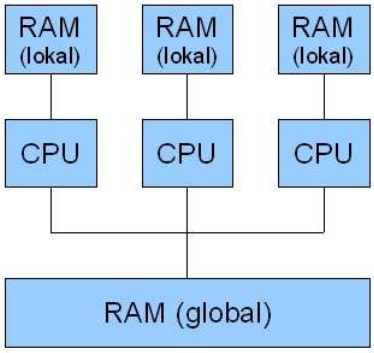

Abbildung 3.3: Nahe Kopplung

3.1.4 Festplatten im Verbund

Beim Zugriff auf externen Speicher, in aller Regel Festplatten, gibt es drei unterschiedliche Konzepte:

- Shared-Everything
- Shared-Nothing
- Shared-Disk

Shared-Everything

Das Shared-Everything-Konzept wird durch einen SMP realisiert. Alle Daten der Festplatten werden im Hauptspeicher abgelegt, auf den alle Prozessoren gleichberechtigt zugreifen können. Dieses Konzept ist jedoch derzeit auf maximal 16 CPUs beschränkt, da die Skalierbarkeit der SMP-Technologie hier derzeit eine Grenze hat.

Shared-Nothing

Das Shared-Nothing-Konzept basiert darauf, dass jedes System einen Teil der Daten exklusiv verwaltet. Die Zuordnung der Arbeitslast der einzelnen Knoten wird meist statisch implementiert. Die statische Aufteilung der Daten hat den positiven Effekt, dass die Synchronisation mit anderen Knoten praktisch entfällt. Als Nachteil ist zu sehen, dass das „Balancing“ der Last nicht dynamisch geregelt ist und im Falle von Last-Schwankungen die Effizienz negativ beeinflusst wird.

Shared-Disk

Im Gegensatz dazu steht das Shared-Disk-Konzept, bei dem jedes System auf den gesamten Datenbestand zugreifen kann. Durch die ungekapselte Verwaltung der Daten kann hier die Last dynamisch verteilt werden. Doch aus diesem Aspekt folgt zugleich auch ein erhöhter Aufwand zur globalen Synchronisation des Datenzugriffs. Zusätzlich entsteht ein „Overhead“ bei der Kohärenzkontrolle, da Daten in mehr als einem Knoten zur selben Zeit vorhanden sein können und in verschiedener Art und Weise modifiziert werden können. Dieser Effekt erschwert die Skalierbarkeit solcher Systeme. Zur besseren Unterscheidung wird diese Art der Koppelung auch als „Shared-Data-Architecture“ bezeichnet [SysPl].
Durch die Realisierung in einer nahe gekoppelten Umgebung, in der die Synchronisation und die Kohärenzkontrolle im globalen Speicher implementiert sind, wird die Skalierbarkeit jedoch entscheidend verbessert, was wiederum einer der Gründe für den Erfolg des Parallel-Sysplex ist [inZos].

3.2 Coupling Facility

In einer nahe gekoppelten Cluster-Architektur wird zwischen dem lokalen Speicher, den jedes System besitzt, und dem globalen Speicher, auf den alle Systeme in einem Cluster gleichberechtigt zugreifen, unterschieden. Der globale Speicher wird von einem eigenem System verwaltet. Dieses System, das die Koppelung realisiert, wird in der Parallel-Sysplex-Umgebung auch Coupling-Facility (CF) genannt.
Die Coupling-Facility ist physisch gesehen ein Bereich im Hauptspeicher, auf den von allen Teilnehmern im Parallel-Sysplex zugegriffen werden kann. Sie besitzt Bereiche für das Locking, das Caching und die List-Structures.

Die CF ist ein eigenständiges System, das einem normalen OS/390 oder z/OS sehr ähnlich ist, jedoch wurde der Coupling Facility ein zusätzlicher Coupling-Facility-Control-Code (CFCC) Microcode hinzugefügt. Mit dem CFCC lassen sich die speziellen Einrichtungen der CF steuern. Die CF bietet die Voraussetzungen, um mehr als zwei Instanzen des Betriebssystems z/OS oder OS/390 über eine sogenannte nahe Kopplung in einem Cluster miteinander zu verbinden. In der Regel wird die Coupling Facility auf einer eigenen logischen Partition (LPAR) eingerichtet. Damit die Systeme eines Parallel-Sysplex ihre Daten miteinander teilen können, benötigen sie eine oder mehrere Verbindungen zu diesem zentralen Knotenpunkt. Diese Verbindungen werden im Allgemeinen durch Glasfaser-Kabel realisiert.

Man unterscheidet grundsätzlich zwischen zwei verschiedene Typen von Verbindungen:

- CFS: Coupling-Facility-Sender ⇒ Verbinden Systeme mit der CF. Diese Kanäle können von mehr als einem System genutzt werden.
- CFR: Coupling-Facility-Receiver⇒ Diese Verbindungen gehen direkt von der CF aus. Im Unterschied zu den CFs müssen sie exklusiv zugewiesen werden, d.h. sie können nicht mehrfach verwendet werden.

Beide Typen müssen im I/O-Definition-Dataset (IODF) definiert werden [sABC5].

Im Allgemeinen wird für einen Sysplex mehr als eine CF definiert. Dies führt zum einen zu einer höheren Ausfallsicherheit und zum anderen zu einer besseren Lastverteilung.

Der Betrieb von einer oder mehreren CFs stellt gewisse Anforderungen an die Umgebung. Zunächst wird ein Prozessor benötigt, der den CFCC verarbeiten kann. Dann muss man mindestens einen Prozessor haben, der ein oder mehrere Betriebsysteme verwalten kann. Als dritte Voraussetzung muss ein Betriebssystem vorhanden sein, um die Ressourcen der CF verwalten zu können.

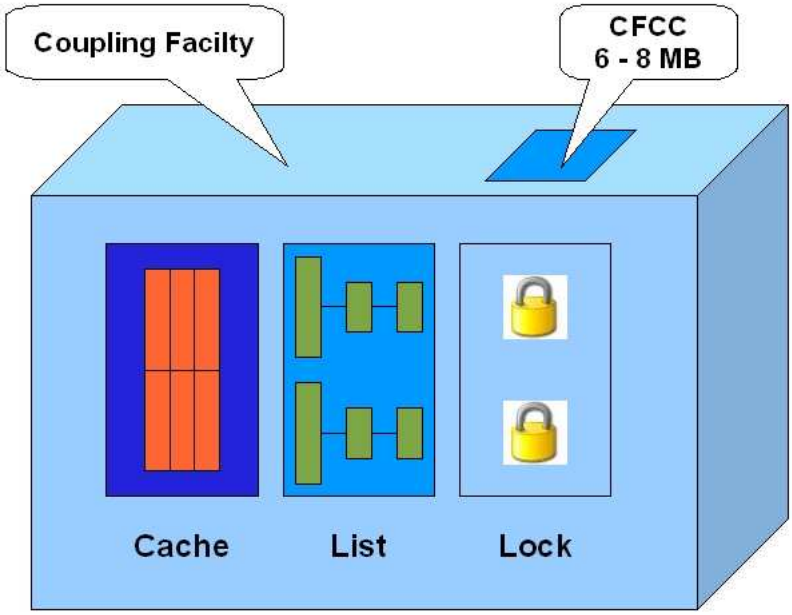

Abbildung 3.4: Struktureller Aufbau einer Coupling Facility

3.3 Strukturen der Coupling-Facility

Der Speicher in der Coupling-Facility wird in Strukturen gegliedert, über die das Highspeed-Datasharing und die Serialisierung durchgeführt werden.

3.3.1 List-Struktur

Die List-Struktur ist notwendig für Anwendungen, die Daten in einem Stapel (Stack, Queue) verarbeiten, und zum Speichern von Statusinformationen über die beteiligten Systeme.

3.3.2 Lock-Struktur

Die Lock-Struktur regelt den exklusiven (read/write) oder den einfachen Zugriff auf Daten sowie die Serialisierung des Zugriffs auf die Daten.

Das Locking von Ressourcen oder Daten verhindert, dass auf die Daten von mehr als einem Prozess zur gleichen Zeit zugegriffen werden kann. Der Locking-Manager serialisiert die Zugriffe auf eine Datenstruktur, in dem er Locks vergibt. Besitzt ein Prozess ein Lock bespielweise auf eine Tabelle, so müssen die anderen Prozesse, die auch auf diese Tabelle zugreifen wollen, warten, bis der aktuelle Prozess das Lock wieder frei gibt.
Durch die zentrale Position der CF im Sysplex und die High-Speed-Verbindungen zu den einzelnen Systemen, bietet die CF optimale Vorraussetzungen für die Skalierbarkeit und eine hohe Performance. Die Vergabe von Locks kann jedoch zu starken Performance-Einbußen führen, wenn Prozesse Daten, die von ihnen nicht verändert werden müssen, trotzdem locken, damit sie nicht von anderen Prozessen verändert werden können. Um diesen Overhead von Locks zu vermeiden, unterschiedet man zwischen den Exclusive-Locks (oder Write-Lock) und den Shared-Locks (oder Read-Lock). Wie die Namensgebung schon andeutet, können Exclusive-Locks nur von einem Prozess gehalten werden, während ein Shared-Lock von mehreren Prozessen zur gleichen Zeit gehalten werden kann. Mit dem Read-Lock wird sichergestellt, dass die Daten, so lange das Lock gehalten wird, nicht verändert werden können.

Nutzung von Locks

In Datenbank- und Transaktions-Management-Systemen sind Locks durch Objekte realisiert, die mindestens vier Methoden haben:

- getReadLock

- getWriteLock
- promoteReadToWrite
- unlock

Transaktionen oder Datenbankzugriffe können im Two-Phase-Commit-Verfahren abgewickelt werden. Es handelt sich dabei um eine Methode, um ein Höchstmaß an Sicherheit beim Zugriff und bei der Verarbeitung von Daten, insbesondere in gekoppelten Systemen bzw. bei konkurrierenden Zugriffen, zu gewährleisten.
Dabei werden in der ersten Phase (die Wachstums-Phase) alle Locks gesetzt und in der zweiten Phase (Schrumpf-Phase) alle Locks wieder freigegeben. Die Phasen sind zeitlich disjunkt, das heißt erst nachdem das letzte Lock erfolgreich gesetzt worden ist, wird der erste Unlock ausgeführt.

Verwaltung von Locks

Zur Verwaltung der Locks gibt es verschiedene Ansätze. Der strukturell einfachste ist die Locking-Table. Ein Datensatz beispielsweise in einer relationalen Datenbank, für den ein Lock gesetzt wurde, implementiert dies bereits. Erfolgt ein Zugriff auf einen solchen Datensatz, wird zuerst der Zustand des Lock-Feldes geprüft. Ist der Datensatz noch nicht gelockt, wird ein Lock beim Locking-Manager beantragt. Erst nach Erhalt des Locks kann der Zugriff auf die Daten erfolgen. Wegen der damit verbundenen Zugriffe auf externen Speicher ist dieses Verfahren wenig performant.
Alternativ zu der Locking-Table gibt es noch den Ansatz des „Invalidate Broadcast“. Nachdem der Prozess einen Datensatz verändert hat, schickt dieser ein „Invalidate Broadcast“ zu allen anderen Prozessen, die ein „Shared-Lock“ halten und signalisiert damit, dass die lokalen Kopien der anderen Prozesse nicht mehr gültig sind. Zeitgleich schreibt der Prozess, der den Datensatz verändert hat, die Daten zurück in den öffentlichen Speicher, damit die anderen Prozesse, die auf diesen Datensatz gewartet haben, die aktuelle Version in den lokalen Speicher laden können. Als Variante dieses Konzeptes besteht die Möglichkeit, dass das Lock vom Prozess gehalten wird, bis ein Konflikt mit einem anderen Prozess auftritt.
Ein Lock kann mit verschiedenen Auflagen behaftet sein, die es ermöglichen, die Vergabe von Locks zu steuern. Ein Interessent eines Locks muss dessen Anforderungen erfüllen, bevor der Interessent das Lock erwerben darf. Beispielsweise kann ein „Shared-Lock“ von mehr als einem Besitzer gehalten werden, während ein „Exclusive-Lock“ nur von einem Besitzer gehalten werden darf. Durch diese Auflagen oder Privilegien können Konflikte

bei der Vergabe von Locks vermieden werden. In einem Parallel-Sysplex ist ein weiterer Ansatz realisiert. Das „Shared-Data-Konzept“ sieht eine Locking-Table im Hauptspeicher der CF vor, wobei jedes der beteiligten Systeme im lokalen Hauptspeicher die Lock-Queues verwaltet.
Der Ablauf eines „Lock-Requests“ ist atomar, das heißt, der anfragende Prozessor wartet solange, bis die Anfrage positiv oder negativ vom Locking-Manager beantwortet wurde. Möglich ist dies durch die geringe Latenz der Verbindungen zwischen System und CF.
Die Locking-Table der CF hat einen simplen Aufbau:
Für jedes Lock existiert ein Eintrag, der die Adresse des haltenden Systems sowie eine 32-Bit-Map, die die Systeme repräsentiert, die ein Interesse an diesem Lock haben, enthält. Jede Anforderung eines Locks wird in der Tabelle vermerkt, in dem ein 19 Byte (152 Bits) langer Name generiert wird. Um Tabellen mit 2^{152} Einträgen zu vermeiden, werden Hash-Klassen angelegt, um die Größe der Tabelle zu reduzieren. Wichtig dabei ist, dass alle beteiligten Systeme den gleichen Hashing-Algorithmus verwenden.
In den teilnehmenden Systemen wird der Speicherbereich für das Locking-Management in folgende vier Bereiche aufgeteilt.

- Local Lock:
  Enthält eine Liste mit den lokalen Locks und der Zustände. Ein Zustand kann folgen Werte haben:
  - 0: keine Kenntnis
  - S: Shared-Lock
  - E: Exclusive-Lock
  - Gx: Ein anderes System hält das Exclusive-Lock
- Zustands- und Queue-Informationen:
  Hier werden Locks aufgelistet, die ohne Beteiligung von anderen Systemen verwaltet werden. Dazu gehören:
  - Lockname
  - Hashklasse
  - Prozess, der das Lock gerade anfordert, der aktuelle Zustand des Locks und der aktuelle Eigentümer
- Local Portion of Global Queues: Stellt die globalen Queues aus Sicht dieses Systems dar, für den Fall, dass ein anderes System zum Global Manager erklärt wird.

- Global Queues: Alle anderen Global Queues [1].

[inZos]

3.3.3 Cache der Coupling-Facility

Ein Cache einer Festplatte soll die Performance-Einbußen bei den vergleichsweise langsamen physikalischen Lese- und Schreib-Zugriffen gering halten. So werden Daten, die erst vor kurzer Zeit abgerufen worden sind, und Daten, die häufig abgerufen werden, in diesem Cache zwischengespeichert. Die verschiedenen Datensätze können abhängig von der Häufigkeit ihrer Anforderungen unterschiedlich lange Verweilzeiten im CF-Cache haben
.
Die Cache-Struktur der CF wird benötigt für das Highspeed-Datasharing sowie die damit verbundene Konsistenzprüfung.
Die Cache-Struktur wird von Subsystemen, die sysplexweit agieren, als globaler Puffer betrachtet. Dabei werden die lokalen Puffer sozusagen zum lokalen Plattenspeicher degradiert. Wenn eine Anwendung durch eine Aktion einen Datensatz verändert, wird nach Abschluss dieser Aktion der Datensatz zurück in den Cache der CF geschrieben. Dieses Update verursacht automatisch ein „Cross-Invalidate-Signal“ (CI) für alle anderen Systeme. Teil des „Cross-Invalidate-Signals“ ist es, die Zeiger auf den veränderten Datensatz zu aktualisieren. Dies wird durch den Link-Prozessor erledigt, welcher eigenständig ohne Unterbrechung des Hauptprozessors arbeitet. Ein zweiter Aspekt der Cache-Struktur innerhalb der CF ist die Nutzung als globaler Hauptspeicher.
Verschiedene Subsysteme, wie DB2 und CICS verwenden die CF als einen „Store-In-Cache“. Die Versionen der Datenstrukturen, die sich auf der CF befinden, können dabei jünger sein als jene, die sich auf den Platten der angeschlosenen Systeme befinden. Sollte die CF einmal ausfallen, so ist es möglich, über die Recovery-Routinen und die Buffer-Pools der einzelnen Systeme den Cache vollständig wieder herzustellen. Ist dies geschehen, wird der Cache auf die Platten zurück geschrieben. Da jedoch die CF keine direkte Verbindungen zu den Festplatten hat, muss diese Aufgabe ein angeschlossenes System übernehmen [inZos].

[1] Zusammen mit dem vorhergegangenen Bereich bilden diese zwei den gesamten globalen Zustand ab

3.4 Nutzer der Strukturen und Policies

Bestimmte Anwendungen, die sysplex-weit agieren, brauchen entsprechende Strukturen auf der CF. Bei den Anwendungen handelt es sich nicht nur um Benutzerprogramme, sondern zum größten Teil um Subsysteme wie RACF, CICS etc..

Es gibt zudem noch einen weiteren Bereich im Speicher, auf den alle Systeme zugreifen können, den sogenannten Dump-Space. Hier werden alle Meldungen der Subsysteme gespeichert. Im Falle eines Fehler muss die Suche danach nicht auf jedem der maximal 32 Systeme ausgeführt werden, sondern kann zentral abgearbeitet werden.

3.4.1 Verwaltung der vorhandenen Strukturen - Policies

Die Verwaltung der vorhandenen Strukturen erfolgt durch eine Coupling-Facility-Resource-Management-Policy (CFRM-Policy). In diesen Richtlinien wird definiert, wie welche Ressourcen zur Verfügung stehen und benutzt werden können. Zu beachten ist dabei, dass zu jedem Zeitpunkt jeweils nur eine Policy aktiv sein kann. Dennoch ist es möglich, mehr als eine Policy zu definieren und je nach Bedarf zu aktivieren. Beispielsweise können je nach Lastanforderung zu verschiedenen Tageszeiten die Richtlinien angepasst und die (jeweils) optimale Policy aktiviert werden. Der Aufbau der Policies erfordert die Angabe jeder verwendeten Struktur und die zu verwendenden Coupling-Facilities. Mit dem Befehl:
`SETXCF START,POLICY,TYPE=CFRM,POLNAME=<policyname>`
wird das System aufgefordert die entsprechende Policy in das CFRM-Dataset zu kopieren und zu aktivieren. Damit wird implizit die vorherige Policy aus dem CFRM-Dataset entfernt und somit deaktiviert.
Um die Coupling-Facility optimal nutzen zu können, ist es nötig, die Subsysteme der beteiligten Betriebssysteme separat in den Parallel-Sysplex zu integrieren. Teilweise brauchen die beteiligten System alle drei der vorhanden Strukturen, teilweise nur die Cache, die List- oder die Locking-Struktur. Diese Subsysteme und Anwendungen werden Exploiter (zu Deutsch: Ausbeuter/Nutzer) genannt.
Zu den Exploitern gehören die folgenden Subsysteme mit ihren benötigten Stukturen (Abb.: 3.5):

JES2	⇒	List
DB2	⇒	Cache, List, Lock
XCF	⇒	List
LOGR	⇒	List
VSAM/RLS	⇒	Cache
ALLOC	⇒	List
VTAM	⇒	List
IMS	⇒	Cache
IRLM	⇒	List, Lock
RACF	⇒	Cache

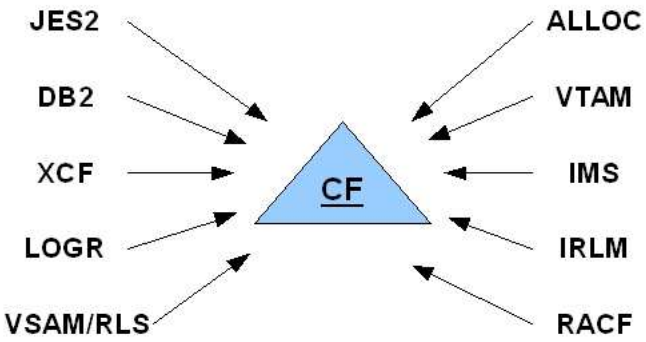

Abbildung 3.5: Exploiter

3.4.2 Couple Datasets

Um einen Parallel-Sysplex betreiben zu können, sind für die beteiligten Systeme Informationen und Richtlinien nötig, nach denen sie agieren sollen. Diese Informationen werden in Form von Policies in den Couple-Datasets (CDS) hinterlegt. Üblicherweise werden zur Gewährleistung der Verfügbarkeit diese CDS in dreifacher Ausführung, das primäre, das sekundäre und das Ersatz-CDS, auf verschiedenen Direct-Access-Storage-Devices (DASD) angelegt. Die DASD, die die CDS beinhalten, sind für alle im Sysplex beteiligten Systeme gleichermaßen zugänglich. Zugleich sind weitere CDS erforderlich, um den Workload-Manager (WLM) und das „Sysplex-Failure-Management“ (SFM) für Cluster mit mehr als einem beteiligtem System betreiben zu können. WLM und SFM hinterlegen Informationen in Form von Policies in den dazugehörigen CDS.
Eine Policy beinhaltet einen Satz an Richtlinien und Vorgaben, die die im Sysplex be-

teiligten Systeme befolgen müssen. Diese Richtlinien können alle Systeme, zu Gruppen zusammengefasste Systeme oder nur einzelne Systeme betreffen. Es ist möglich, für verschiedene Szenarien entsprechende Richtlinien zu erstellen und diese nach Bedarf einzusetzen.
Die verschiedenen CDS werden je nach Bedarf für die entsprechenden Policies bereits bei der Planung des Parallel-Sysplex angelegt. Man unterscheidet sechs Arten von CDS:

- ARM ⇒ Automatic Restart Management: Legt fest, wie mit abgebrochenen und deshalb neu zu startenden Tasks und Jobs zu verfahren ist.
- LOGR ⇒ System Logger
- CFRM ⇒ Coupling Facility Resource Management: legt fest, wie die Ressourcen der Coupling Facility zu verwalten sind.
- SFM ⇒ Sysplex Failure Management: regelt das Verfahren bei Konnektivitätsfehlern.
- WLM ⇒ Workload Manager: Über die entsprechenden Policies werden Ziele für bestimmte Lastprofile definiert und in Abhängigkeit davon der Sysplex gesteuert.
- XCF ⇒ Cross Coupling Facility:

Jede Richtlinie ist, sobald sie aktiv ist, abhängig von Real-Time-Informationen und den Ressourcen des Parallel-Sysplexes.
[sABC5]

3.5 Dimensionieren der Strukturen

3.5.1 CF-Sizer

IBM stellt mit dem Tool „CF-Sizer“ ein Web-basiertes Werkzeug zur Verfügung, das bei der Erstellung und Skalierung der Strukturen innerhalb der Policy hilfreiche Informationen liefert.
Die Bedienung dieses Tools ist einfach, da die Informationen, die benötigt werden, um die Größe der zu erstellenden Struktur zu berechen, gut struktriert dargestellt werden. Zu jedem Exploiter wird über die jeweilige <Exploiter>-Hilfe eine genaue Beschreibung geliefert, wie und wo die Ausgangsdaten zu erhalten sind, und in welcher Form sie in die Eingabemaske einzugeben sind.

Nach ihrer Eingabe wird durch Klicken auf den „Submit“-Button die Größe der zu erstellenden Struktur berechnet. Die Ausgabe des Ergebnisses bietet neben den nackten Zahlen auch weitere Informationen, wie Name der Struktur, falls dieser nicht frei wählbar ist, initiale Größe und Gesamtgröße.
Des Weiteren wird anhand von einem Pseudo-Code-Beispiel gezeigt, wie der Code innerhalb des „Formatierungs-Jobs“ auszusehen hat [CFSiz].

3.5.2 Anlegen von Strukturen mit Hilfe von CF-Sizer

Um eine Struktur innerhalb einer Richtlinie (Policy) anzulegen und zu skalieren, wurde der Job 'IXCCFRM' stückweise erweitert. Nachdem die Policy durch eine Struktur ergänzt und aktiviert worden ist, muss dem entsprechenden Exploiter noch mitgeteilt werden, dass jetzt eine Struktur in der Policy existiert, die sein Verhalten beinflusst. Diese Mitteilung kann entweder im laufenden Betrieb durch einen entsprechenden Befehl oder durch einen Neustart der Systeme übermittelt werden.

Nachfolgend wird erklärt, wie die Strukturen für die einzelnen Exploiter mit Hilfe des CF-Sizers angelegt werden:

XCF

Durch das Aktivieren der List-Strukturen für die Cross-Coupling-Facility-Datasets (XCF-Datasets) entstehen signifikante Vorteile beim Recovery-Management und der Verfügbarkeit der einzelnen Systeme [CFSiz].
Dabei nutzt XCF die List-Struktur der CF. Um die Policy entsprechend zu erweitern, sind in der Maske des CF-Sizers zwei Angaben zu machen:

1: **CLASSLEN**
   Die CLASSLEN ist die Größe der Transportklasse, die der Administrator zuvor definiert hat. Per Default ist CLASSLEN auf 956 gesetzt, es können aber Werte von 1 bis 62464 gesetzt werden. Durch diesen Wert wird die Größe jener Nachrichten bestimmt, für die die Transportklasse optimiert wird. Den Wert, den der Administrator angegeben hat, findet man im COUPLExx-Member der 'SYS1.PARMLIB' im 'CLASSDEF'-Statement.
   Die Klassen können von System zu System unterschiedlich sein. Um die richtige Größe der Struktur ermtteln zu können, sollte die kleinste CLASSLEN innerhalb

der Systeme in die Maske des CF-Sizers eingetragen werden. In der hier beschriebenen Installation werden die COUPLESx Member der 'SYS1.PARMLIB' mit identischer CLASSLEN beschrieben.

2: **# of Systems**
Die Anzahl der im Sysplex beteiligten Systeme.

Das Ergebnis der Kalkulation vom CF-Sizer liefert nun eine detailierte Beschreibung, wie die Struktur im Job 'IXCCFRM' zu erweitern ist. Es werden Vorgaben gemacht, wie die Struktur heißen muss und welche Größe sie haben soll (siehe Listing 3.1 auf Seite 37).

Listing 3.1: Auszug aus IXCFRM: Anlegen der XCF Struktur

```
STRUCTURE NAME(IXCLST01)
          SIZE(8192)
          INITSIZE(8192)
          PREFLIST(CF01,CF02)
          ENFORCEORDER(YES)

STRUCTURE NAME(IXCLST02)
          SIZE(8192)
          INITSIZE(8192)
          PREFLIST(CF02,CF01)
          ENFORCEORDER(YES)
```

JES2

JES2 benutzt die List-Struktur der CF für seine Checkpoint-Datasets, die zwei unabhängige Funktionen haben:

1: **Queuing**
Job- und Output-Backup-Queuing, um den Restart zu erleichtern

2: **MAS**
Multi-Access-Spool zur Lastverteilung unter den beteiligten Systemen und zur Gewährleistung von effizienten und unabhängigen Operationen

Die Funktion der Checkpoint-Datasets hängt davon ab, ob es sich um einen Single-Member-MAS oder einen Multi-Member-MAS handelt.
Beim Single-Member-MAS gibt es zwar eine Multi-Access-Spool-Einrichtung, diese wird aber nur von einem System genutzt.

Dagegen gibt es beim Multi-Member-MAS Performance-Vorteile und bessere Zugriffsmöglichkeiten für alle Systeme im Sysplex, wenn die primären Checkpoint-Datasets auf der CF liegen. Zudem wird durch die Locking-Mechanismen der CF die Serialisierung des Zugriffs auf die Datasets sehr effizient geregelt.
Durch die Implementierung des First-In-First-Out-Prinzips (FIFO-Prinzip) im Locking-Mechanismus wird nämlich gewährleistet, dass alle Mitglieder des MAS gleichberechtigt auf die Checkpoint-Datasets zugreifen können.

Um die Struktur ensprechend zu skalieren, muss in der ensprechenden Maske des CF-Sizers die Anzahl der 4 kByte großen Checkpoint-Records eingetragen werden [CFSiz]. Nach der Berechnung wird der Job IXCFRM um folgende Zeilen erweitert:

Listing 3.2: Auszug aus IXCFRM: Anlegen der JES2 Struktur

```
STRUCTURE NAME(JESCKPT1)
          SIZE(2048)
          PREFLIST(CF02,CF01)
          REBUILDPERCENT(1)
```

OPERLOG

OPERLOG ist ein Teil der System-Logging-Serices zu dem auch der LOGREC-Service gehört. Jedoch können diese beiden Services unabhängig voneinander betrieben werden. OPERLOG erlaubt Anwendungen, Daten zu schreiben, zu verändern oder zu löschen. Dabei werden die List-Strukturen der CF genutzt. Dadurch können Anwendungen auf verschiedenen Systemen sysplex-weit miteinander kommunizieren [CFSiz].

Damit der CF-Sizer eine Skalierung der Struktur durchführen kann, sind zwei Eingaben zu machen:

1: **Writes Per Second**
   Um die Einträge pro Sekunde zu ermitteln, sollte über einen deutlich größeren Zeitraum als eine Sekunde hinweg diese Zahl gemessen werden. Dann wird der auf eine Sekunde heruntergerechnete Wert in die Maske des CF-Sizers eingetragen.

2: **Offload Time**
   Mit dieser Größe wird die Verweilzeit der Records in der Cache-Struktur der Coupling Facility angegeben. Nach Ablauf dieser Zeit werden die Daten in einen lokalen Speicher ausgelagert.

Die CF-Sizer Berechnungen führen zu folgenden Zeilen im Job 'IXCFRM':

Listing 3.3: Auszug aus IXCFRM: Anlegen der OPERLOG Struktur

```
STRUCTURE NAME(OPERLOG)
    SIZE(10000)
    PREFLIST(CF02,CF01)
    REBUILDPERCENT(1)
```

LOGREC

LOGREC ist ähnlich aufgebaut wie der OPERLOG. Entsprechend wird auch die Struktur im IXCFRM-Job ähnlich ergänzt:

Listing 3.4: Auszug aus IXCFRM: Anlegen der LOGREC Struktur

```
STRUCTURE NAME(LOGREC)
    SIZE(10000)
    PREFLIST(CF01,CF02)
    REBUILDPERCENT(1)
```

RACF

Das IBM-Sicherheitsssystem RACF benutzt die CF als Cache für häufig angeforderte Daten. Durch die Nutzung einer RACF-Datenbank auf der CF sind die dort gespeicherten Daten schnell für jedes System zugänglich.
Pro RACF-Datenbank werden zwei Strukturen in der CF benötigt, eine für die primäre Datenbank und eine für die Backup-Datenbank. Die Struktur der Backup-Datenbank benötigt nur etwa 20% des Platzes der primären Datenbank.
Die Mindestgröße für die primäre Struktur muss so groß gewählt werden, dass alle Daten, die von den internen I/O-Puffern geliefert werden, aufgenommen werden können [CFSiz].
Die Maske des CF-Sizers fordert vom Benutzer folgende Eingaben, damit ein Vorschlag für die Skalierung der Struktur errechnet werden kann:

1: **Max # Instorage Local Buffers**
   Anzahl der Puffer, die im Sysplex definiert sind. Maximal können 255 Puffer zugeordnet werden. Von IBM wird die Maximalanzahl als Eingabe empfohlen.

2: **# Additional Buffers**
   Anzahl der zusätzlichen 4 kByte Puffer, die die Caching-Funktion verbessern sollen. Als Abschätzung kann man sich an der Anzahl der Profile orientieren, die häufig gebraucht werden.

3: # **MVS-Images**
Anzahl der Systeme, die sich die gleiche RACF-Datenbank teilen.

Nach Eingabe der Werte wird der Job IXCFRM mit folgenden Zeilen ergänzt:

Listing 3.5: Auszug aus IXCFRM: Anlegen der RACF Struktur

```
STRUCTURE NAME(IRRXCF00_P001)
          SIZE(5120)
          PREFLIST(CF01,CF02)
          REBUILDPERCENT(1)

STRUCTURE NAME(IRRXCF00_B001)
          SIZE(2048)
          PREFLIST(CF02,CF01)
          REBUILDPERCENT(1)
```

GRS

GRS (Global Resource Serialisation) nutzt die Locking-Struktur der CF, um den Zugriff der einzelnen Systeme auf die globalen Ressourcen zu serialisieren. Dabei wird festgestellt, welches System Zugriffe für welche Ressource beansprucht [CFSiz].
Damit die Struktur geplant werden kann, sind folgende Angabe nötig:

1: **Peak # Global Resources**
Maximalanzahl der globalen Ressourcen = der Anzahl global gemanagter Ressourcen zum Zeitpunkt der größten Last.

1: **Max # In-Storage Local Buffers**
Maximale Anzahl der im Sysplex beteiligten Systeme.

Das Ergebnis des CF-Sizers wird verwendet, um folgende Zeilen in den Job IXCFRM einzufügen:

Listing 3.6: Auszug aus IXCFRM: Anlegen der GRS Struktur

```
STRUCTURE NAME(ISGLOCK)
          SIZE(2560)
          PREFLIST(CF01,CF02)
          ENFORCEORDER(YES)
```

Weitere Exploiter

Bei der hier zu Grunde liegenden Aufgabenstellung war es nicht nötig, alle z/OS-Subsysteme in den Sysplex zu integrieren. Weitere Subsysteme, die über die CF miteinander kommunizieren können sind [CFSiz]:

BatchPipes	DB2
CICS	CICS-Log-and-Journal
DFSMShsm Common-Recall-Queue	Enhanced-Catalog-Sharing (ECS)
Health-Checker	IMS
IRD	MQ-Series
OEM-Cache	OEM-Lock
Tape-Devices	TCP/IP Sysplexreports
TCP/IP Sysplex-wide Security Associations (SWSA)	RRS
SMF	VTAM
VTAM-RLS	WLM

ECS ist nur möglich für Systeme, die nicht als Gast unter einem z/VM System betrieben werden

4 Allgemeines zum Cloning von z/OS-Instanzen

Bei dem im Mainframe-Umfeld mit Klonen bezeichneten Verfahren handelt es sich um eine Methode, bei der mindestens eine zweite Instanz des Betriebsystems z/OS erstellt wird. Beide Instanzen teilen sich den überwiegenden Teil der für den Betrieb wichtigen Dateien (z.B. Programm- und Konfigurations-Bibliotheken). Dieses Teilen („SHARING“) ist möglich, da auf diese Daten praktisch nur im „read-only“-Modus zugegriffen wird. Nur für Wartungs- bzw. Konfigurationsanpassungen greifen besonders autorisierte Systemspezialisten ändernd (Write, Update, Alter, Delete) auf derartige Dateien zu. Lediglich die 3390-Festplatten, auf denen die Paging- und die Spooling-Datasets betrieben werden, müssen für jede z/OS-Instanz alleine vorhanden sein.

Für ein „normales“ z/OS System werden üblicherweise sechs 3390 Modell 3 Platten á 2,8 GigaByte benötigt. Auf vier dieser Platten erfolgt ein Lese-Zugriff, auf die verbleibenden Platten wird lesend und schreibend zugegriffen (Abb.:4.1 auf Seite 43). Es liegt also nahe, die Platten, auf die nur lesend zugegriffen wird, für alle Systeme dieser Bauart genau einmal gemeinsam bereit zu stellen und den Lesezugriff für mehr als ein System zu erlauben.

Konkret bedeutet Klonen:

- **Format**: Formatieren der benötigten Platten
- **Build**: Einrichten der Infrastruktur auf den Platten
  - Definieren, Erzeugen
  - Kopieren aus bzw. Zugreifen auf vorhandene Infrastruktur
- **Tayloring**: Anpassen, Zuschneidern, Ändern der Konfigurations-Dateien
- **Iterate**: Wiederholen der Schritte für weitere Klone

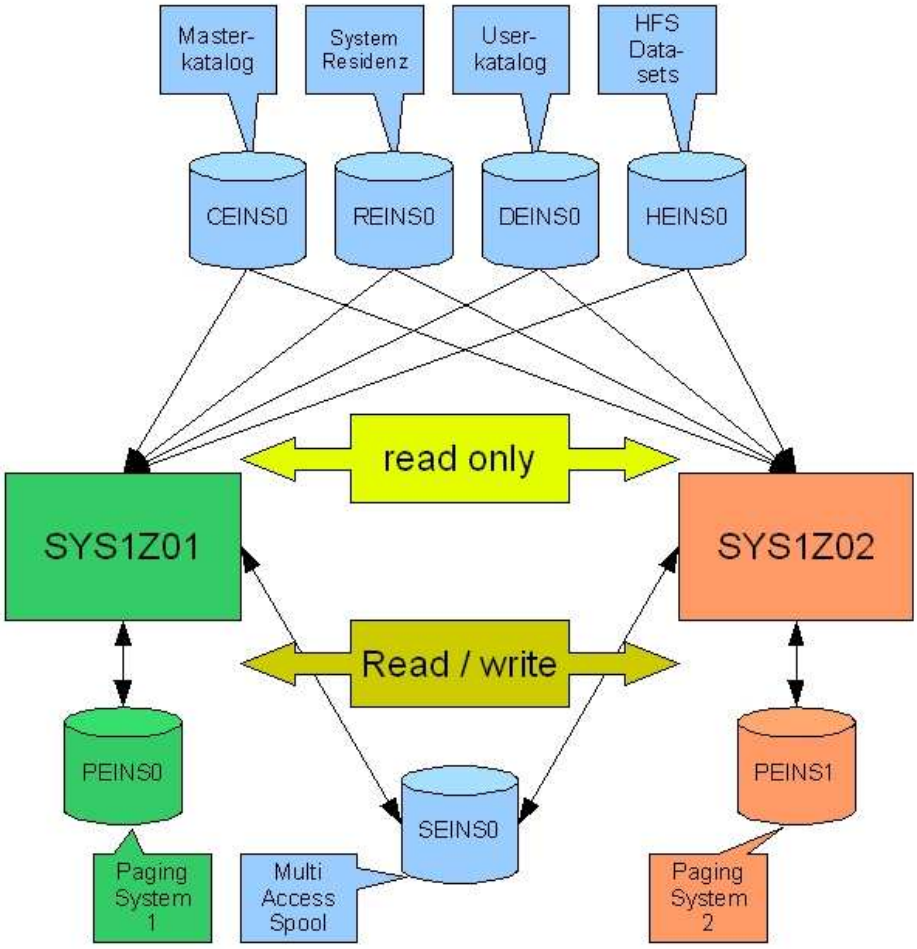

Abbildung 4.1: Prinzip des Klonens

4.1 Zu beachtende Rand-Bedingungen

4.1.1 Datei-Organisation und Zugriffsmethoden

Auf IBM-Betriebssystemen kommen verschiedene Datei-Organisationen und entsprechende Zugriffs-Methoden zum Einsatz, was Auswirkungen auf die folgenden Aktivitäten haben wird.

VSAM

Dateien werden in z/OS vielfach mit der Zugriffsmethode VSAM verwaltet und bearbeitet. VSAM bedeutet Virtual Storage Access Method. Zentraler Punkt dieser Dateiver-

waltung und Zugriffsmethode ist das **Katalogmanagement**. Über einen sogenannten **Master-Katalog** bzw. über darunter liegende Benutzer-/**User-Kataloge** wird fast alles gesteuert. VSAM-Dateien werden mit den speziellen Dateiattributen in einem dieser Kataloge gespeichert und stehen damit für alle Anwendungen bereit. Im Katalog befindet sich auch die Information, wo auf welcher physikalischen Platte oder auf welchem Band-/Kassetten-Laufwerk eine Datei gespeichert ist. Entwickler oder Arbeitsvorbereiter müssen also nicht unbedingt wissen, wo Dateien wirklich gespeichert sind. Daher benötigt jedes z/OS-System **einen Master-Katalog** und normalerweise einen oder mehrere **User-Kataloge** [JaMos].

Neben VSAM spielen auch noch **andere Zugriffsmethoden** eine wichtige Rolle. Dies sind neben den zwei sequentiellen Zugriffarten QSAM (Queued Sequential Access Method) bzw. BSAM (Basic Sequential Access Method) noch BPAM (Basic Partitioned Access Method).

QSAM/BSAM

Bei QSAM/BSAM werden die **Datensätze in Zugangsfolge** in den entsprechenden Dateien gespeichert und können auch nur in dieser Reihenfolge wieder gelesen werden.

BPAM

BPAM-Dateien sind daran zu erkennen, dass in ihnen sogenannte **Member** gespeichert werden und dass man sie häufig als **Bibliothek (Library)** bezeichnet. Üblicherweise enthalten sie Programme, Source-Code, Configurations-Members, Listen etc., die meistens logisch zusammen gehören. Der Zugriff auf ein Member erfolgt über ein BPAM-Inhaltsverzeichnis, das **Directory**.

Die übrigen IBM-typischen Datei-Formen sollen hier nicht weiter erläutert werden, da sie im Rahmen dieser Arbeit kaum eine Rolle spielen.

Katalogisierung

Um die Möglichkeiten der Katalog-gestützten Datei-Verwaltung auch für Nicht-VSAM-Zugriffs-Methoden (wie BPAM, QSAM etc.) zu nutzen, kann man diese Dateien als sogenannte Non-VSAM-Objekte (NVSAM) in den Katalogen registrieren. Diesen Vorgang nennt man auch **katalogisieren**. Erzeugt werden solche Dateien mit den ihnen eigenen Methoden; danach werden Sie 'katalogisiert' und erlauben dann den Zugriff über einen

VSAM-Katalog.

4.1.2 Platten-Layout

Aus Gründen der Ausfallsicherheit, zur Vermeidung eines „Single-Point-Of-Failure“ und zur Steigerung der Performance werden der Master-Katalog und die User-Kataloge auf verschiedene Platten abgelegt, genauso wie einige sogenannte SYS1-Datasets (deren Namen beginnen mit 'SYS1') separat von anderen wichtigen Daten angelegt werden. Die Platten für das erste System haben die folgenden Spezifikationen:

- **CEINS0**: C-Platte = Anlage des Master-Katalogs
- **DEINS0**: D-Platte = Anlage der User-Kataloge
- **HEINS0**: H-Platte = Anlage des Hierarchical-File-Systems (HFS)
- **REINS0**: R-Platte = Residenz der betriebsystemspezifischen Daten
- **PEINS0**: P-Platte = Page-Datasets-Platte für jedes System exklusiv
- **SEINS0**: S-Platte = Spool- und Dump-Platte für jedes System exklusiv

Die Platten „PEINS0“ und „SEINS0“ werden für das erste Betriebssystem exklusiv angelegt, und können auch nur von diesem schreibend verändert werden. Darum werden für ein weiteres System wiederum zwei Platten im exklusiven Modus für das Paging und Spooling benötigt, die entsprechend der Konvention zur Vergabe der Namen der Platten „PEINS1“ und „SEINS1“ heißen werden.

Anmerkung:
Sofern beim Spooling das **Multi-Access-Spool-Verfahren** (MAS-Verfahren) zum Einsatz kommt, ist die Platte SEINS0 nicht für jedes System exclusiv vorhanden, sondern wird im READ/WRITE-Modus zwischen den beteiligten Systemen „geshared“. Für dieses Projekt wurde das MAS-Verfahren eingesetzt. Es wird weiter unten auf den Seiten 61 und 78 kurz beschrieben.

4.1.3 Namenskonventionen

Besondere Namenskonventionen sind gerade bei der Systemkonfiguration in einem Sysplex notwendig, um die Administration zu vereinheitlichen und zu unterstützen. Daneben spielen aber auch Sicherheitsaspekte beim Benennen von Dateien oder anderen Ressourcen

eine wichtige Rolle. Ohne klare Regeln für z.B. das Benennen von Dateien wird die Rechteverwaltung mit RACF (Resource Access Control Facility) ziemlich unübersichtlich, erfordert hohen zeitlichen Aufwand und wird zunehmend komplexer [DSN 1].

Datei-Bezeichnungen

Beim Benennen von Dateien sind z/OS-bedingt bestimmte Regeln einzuhalten [DSN 1]:

- **Dateinamen** unter z/OS dürfen **nicht länger als 44 Stellen** sein (Ausnahme HFS-Dateien - siehe unten).
- Man kann **alle Alphazeichen, die Ziffern und die Sonderzeichen # (Hash), @ (At), $ (Dollar)** verwenden.
- **Nach spätestens 8 Zeichen** muss ein Punkt folgen.
- **Zwei aufeinander folgende Punkte** oder ein **Ende des Namens mit einem Punkt** sind nicht erlaubt.

Die Zeichenfolge bis zum ersten Punkt bezeichnet man in diesem Zusammenhang als **High-Level-Qualifier (HLQ)**, die ggf. weiteren Sequenzen sind Second-Level-Qualifier, Mid-Level-Qualifier oder ganz rechts der Low-Level-Qualifier.
Mit Hilfe dieser Qualifier insbesondere aber basierend auf dem HLQ lassen sich beispielsweise in **RACF sogenannte Dataset-Profile**

- in **generischer Form** (z.B. `SYS1.**, SYS1.EINS.*`, etc.) oder
- in **diskreter(er)** Form (z.B. `SYS1.PARMLIB oder HSF.EINS.%%%`, etc.)

anlegen, die für Dateien mit entsprechendem Namens-Aufbau eine genau bestimmte Rechtevergabe ermöglichen.

Beim Klonen spielen solche Überlegungen auch eine wichtige Rolle:

- Für jedes System benötigte Dateien sollten den Hiqh-Level-Qualifier „SYS1“ haben.
- Sofern Dateien spezifisch für jedes einzelne System anzulegen sind, wird als High-Level-Qualifier der Systemnamen, also „EINS“, „ZWEI“ etc. benutzt.
- Dateien, die sysplex-spezifisch sind, erhalten den Namen des Sysplex's als High-Level-Qualifier also hier „STORPLEX“.

- Generell sollte die Namensvergabe bei den Dateien so sein, dass über die Qualifier Gruppen von Anwendungen oder Subsystemen gebildet werden, damit die Dateien einerseits in den Datei-Übersichten zusammenhängend erscheinen (z.B. alle mit HSF.EINS beginnenden Dateien) und andererseits die Rechtevergabe über diese Gruppen-Bezeichnungen (s.o. HSF.EINS.** qualifiziert alle so beginnenden Dataset-Namen) erfolgen kann.

Nicht immer lassen sich derartige Regeln konsequent einhalten. So werden beispielsweise die später beschriebenen BROADCAST- und STGINDEX- und UADS-Datasets, die den High-Level-Qualifier „SYS1“ tragen, jedoch system-spezifische Datasets sind, mit dem Second-Level-Qualifier „EINS“ oder „ZWEI“ ausgestattet. Das gilt auch für Dateien, die zum Nucleus gehören und daneben auch systemspezifische Informationen enthalten.

4.2 Ausgangs-Situation

Im folgenden Abschnitt wird davon ausgegangen, dass bereits ein lauffähiges z/OS-System existiert, von dem die systemrelevanten Daten kopiert werden können. Dieses initiale System dient als Quelle für das erste Klon-System. Alle weiteren Klone werden vom ersten Klon erzeugt.

Auf einen Teil der Dateien des Ursprungssystems werden weiter unten Zugriffsmöglichkeiten ermöglicht, in dem die fraglichen Dateien in den Master-Katalog des geklonten Systems mit Referenz auf das 'alte' System (Referenz ist der Plattenname [VolSer]) eingetragen werden. Das Nutzen von Teilen der vorhandenen Infrastruktur macht Sinn, weil so unnötige Redundanzen vermieden und diese Dateien im Rahmen der Wartung des 'Alt-Systems' auf einem aktuellen Stand gehalten werden. Diesen Umstand muss man berücksichtigen, wenn man das Klon-Verfahren auf fremde System-Komplexe überträgt.

5 Erzeugen des Master-Klons

5.1 Vorgehensweise

5.1.1 Step 1 Benötigte Platten bereitstellen und formatieren

Im ersten Schritt werden die sechs benötigten Platten von eventuell vorhandenen Daten befreit, indem sie neu formatiert werden. Dazu müssen die entsprechenden Platten vom System getrennt werden. Dies geschieht mit dem folgenden Befehl:

`'VARY <VOLSER-Bereich>,OFFLINE'`

Nun können mit dem Programm „ICKDSF“ (ein IBM-Initialisierungs-Utility) die Platten re-initialisiert werden. Bei diesem Vorgang werden entsprechend der Namenskonvention die Platten mit den jeweiligen Volume-IDs bezeichnet (5.1).

Listing 5.1: JOB001 Aufbau VTOC + Index

```
//INITDSK  EXEC PGM=ICKDSF
//*-------------------------------------------------------------------*
//*          CREATE VTOC & INDEX                                      *
//*-------------------------------------------------------------------*
//*    ATTENTION CHECK DEVICE-NUMBERS                                 ***
//*    FIRST: SET DEVICES  OFFLINE (VARY 300-306,OFFLINE)             ***
//*    SECON: RUN JOB001, ANSWER REQUESTS WITH 'U'                    ***
//*    THIRD: SET DEVICES ONLINE (VARY 300-306,ONLINE)                ***
//*-------------------------------------------------------------------*
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
 INIT NOCHECK NOVERIFY NOVALIDATE PURGE VTOC(0,3,12) INDEX(0,1,2) -
      UNIT(300) VOLID(REINS0)
 INIT NOCHECK NOVERIFY NOVALIDATE PURGE VTOC(0,3,12) INDEX(0,1,2) -
      UNIT(301) VOLID(CEINS0)
 INIT NOCHECK NOVERIFY NOVALIDATE PURGE VTOC(0,3,12) INDEX(0,1,2) -
      UNIT(302) VOLID(DEINS0)
 INIT NOCHECK NOVERIFY NOVALIDATE PURGE VTOC(0,3,12) INDEX(0,1,2) -
      UNIT(303) VOLID(PEINS0)
 INIT NOCHECK NOVERIFY NOVALIDATE PURGE VTOC(0,3,12) INDEX(0,1,2) -
```

```
      UNIT(304)  VOLID(HEINS0)
 INIT NOCHECK NOVERIFY NOVALIDATE PURGE VTOC(0,3,12) INDEX(0,1,2) -
      UNIT(305)  VOLID(SEINS0)
```

5.1.2 Step 2 Master-Katalog einrichten

Der nächste Schritt beinhaltet die Erstellung des Master-Katalogs, auf der dafür vorgesehen Platte CEINS0. Dazu wird zunächst ein VVDS-Dataset (VSAM Volume Data Set) angelegt, in das später u.a der Name des Master-Katalogs eingetragen wird.

Ein VVDS enthält zusätzliche Katalog-Informationen über VSAM Datasets, „Non-VSAM-Datasets“ und SMS-verwaltete Datasets, die auf dem gleichen Volume wie das VVDS angelegt sind (Erläuterung von SMS: siehe 5.1.5 auf Seite 54). Jedes Volume, das ein VSAM- oder SMS-verwaltetes Dataset enthält, benötigt zwingend ein VVDS. Das VVDS agiert für verschiedene Typen von Datasets als Katalog-Ergänzung und VTOC-Erweiterung (VTOC steht für Volume Table of Contents oder Datenträgerinhaltsverzeichnis). Ein VVDS kann Informationen über Datasets, die bereits katalogisiert sind, enthalten [zABC3]. Im Master-Katalog müssen alle weiteren User-Kataloge inklusive der Aliase gespeichert werden. Aliase sind zusätzliche Namen für Datasets, unter denen diese erreicht werden können oder auch Zeiger z.B. auf einen Benutzerkatalog.

Alias als zusätzlicher Namen für eine Datei:
Z.B. gibt es bei der Programmgenerierung im System drei Bibliotheken, die heißen:

```
SYS1.CEE.V1.SCEELINK
SYS1.CEE.V2.SCEELINK
SYS1.CEE.V3.SCEELINK
```

Diese drei Dateien enthalten unterschiedliche Wartungsstände der SCEELINK-Library. Die Entwickler benötigen zur Programm-Generierung eine SCEELINK-Bibliothek. Da man ihnen nicht zumuten kann, mal auf Wartungsstand 1, 2 oder 3 zuzugreifen, richtet man systemtechnisch einen Alias ein, der auf die zu nutzende Bibliothek zeigt:

```
DEFINE ALIAS ( NAME(SYS1.COBOL.SCEELINK) RELATED(‘SYS1.COBOL.V2.SCEELINK‘))
```

Damit zeigt der Alias `‘SYS1.COBOL.SCEELINK‘` jetzt auf die physikalische Datei `‘SYS1.COBOL.V2.SCEELINK‘`. Änderungen sind jederzeit möglich, ohne dass die Entwickler an ihren Jobs oder Prozeduren eine Änderung vornehmen müssen, weil diese nur über

den Alias auf die gewünschte Datei zugreifen.

Alias als Zeiger auf einen Katalog:
Aliase werden daneben auch benutzt, um z.B. die Dateien von Benutzern auf bestimmte Kataloge, Festplatten etc. zu lenken:
`DEFINE ALIAS ( NAME(USER999) RELATED('USERCAT.FOR.ALL.USERS'))`

Der High-Level-Qualifier `'USER999'` wirkt hier als Zeiger auf den Benutzerkatalog `'USERCAT.FOR.ALL.USERS'`.

Vor der Definition eines neuen Master-Katalogs muss man sicherstellen, dass noch kein Katalog existiert. Darum startet der Job mit einer entsprechenden Löschanweisung. Falls kein Katalog vorhanden ist, gibt es bei der entsprechenden Anweisung ein Condition-Code von 12, was akzeptiert werden kann. Jetzt wird ein neuer Master-Katalog definiert und ein Alias dafür erzeugt, damit Einträge in den Master-Katalog über diesen Alias getätigt werden können.

Listing 5.2: JOB002 Aufbau VVDS-CEINS0 + Master-Katalog

```
//EXPORT1 EXEC PGM=IDCAMS
//*******************************************************************
//*         DELETE OLD DATA,  EXPECT RC=12                          *
//*******************************************************************
//SYSPRINT  DD SYSOUT=*
//CAT       DD DISP=SHR,UNIT=3390,VOL=SER=CEINS0
//SYSIN     DD *
  EXPORT   M.CEINS0  DISCONNECT
//*
//DEFCAT2 EXEC PGM=IDCAMS
//*******************************************************************
//*         DEFINE VVDS,  MCAT                                      *
//*******************************************************************
//SYSPRINT  DD SYSOUT=*
//CAT       DD DISP=SHR,UNIT=3390,VOL=SER=CEINS0
//SYSIN     DD *
  DEFINE   CLUSTER  ( NAME(SYS1.VVDS.VCEINS0) VOLUME(CEINS0) +
                              CYL(1  0) NONINDEXED )
  DEFINE   MASTERCATALOG  ( NAME(M.CEINS0)      VOLUME(CEINS0) FILE(CAT) +
                              STORCLAS(NONSMS) CYL(5  1) ICFCATALOG ) +
                        DATA  ( CYL(3  1)  ) INDEX  ( CYL(2  1)  )
  DEFINE   ALIAS(NAME(EINSM  ) REL(M.CEINS0))
```

5.1.3 Step 3 User-Katalog(e) einrichten

Das Erstellen eines User-Katalogs ähnelt der Vorgehensweise beim Master-Katalog. Zuerst wird auch hier der möglicherweise vorhandene Katalog gelöscht. Sollte das Programm IDCAMS (VSAM-Utility) keinen Katalog zum Löschen finden, so wird ein Condition-Code von 12 zurückgegeben. Danach wird auf der Platte DEINS0 ein VVDS und ein User-Katalog angelegt. Natürlich müssen alle Kataloge und deren Aliase im Master-Katalog gespeichert sein. Dazu wird mit dem „IMPORT CONNECT“-Kommando der User-Katalog mit dem Master-Katalog verbunden.

Listing 5.3: JOB003 Aufbau VVDS-DEINS0 + User-Katalog

```
//ULISTORC JOB (ACCOUNT),'ULISTOR',MSGCLASS=X,MSGLEVEL=(1,1),
//            NOTIFY=&SYSUID,CLASS=A,REGION=6M
//*****************************************************************
//*    JOB  SUBMITTED FROM ULISTOR.SYSPLEX.CNTL(JOB003)            ***
//*    DOC: DEFINE VVDS & USERCATALOG                              ***
//*****************************************************************
//*----------------------------------------------------------------
//DEFCAT1 EXEC PGM=IDCAMS
//*****************************************************************
//*         DEFINE VVDS, UCAT                                      *
//*****************************************************************
//SYSPRINT  DD SYSOUT=*
//CAT       DD DISP=SHR,UNIT=3390,VOL=SER=DEINS0
//SYSIN     DD *
  EXPORT  U.DEINS0 DISCONNECT
  DEFINE  CLUSTER ( NAME(SYS1.VVDS.VDEINS0) VOLUME(DEINS0) +
                    CYL(1 0) NONINDEXED )
  DEFINE    USERCATALOG ( NAME(U.DEINS0)    VOLUME(DEINS0) +
                    STORCLAS(NONSMS)      +
                    FILE(CAT) CYL(3 1) ICFCATALOG ) +
                DATA ( CYL(1 1) ) +
               INDEX ( CYL(1 1) )
  IMPORT  CONNECT OBJECTS((U.DEINS0 DEVT(3390) VOL(DEINS0))) +
                CATALOG ( M.CEINS0 )
  DEFINE  ALIAS(NAME(SYSEINS) REL(U.DEINS0)) CATALOG(M.CEINS0)
  DEFINE  ALIAS(NAME(SMP0 )   REL(U.DEINS0)) CATALOG(M.CEINS0)
  DEFINE  ALIAS(NAME(IODF )   REL(U.DEINS0)) CATALOG(M.CEINS0)
  DEFINE  ALIAS(NAME(SMS  )   REL(U.DEINS0)) CATALOG(M.CEINS0)
  ...
  DEFINE  ALIAS(NAME(IBMUSER) REL(U.DEINS0)) CATALOG(M.CEINS0)
  DEFINE  ALIAS(NAME(EINSU )  REL(U.DEINS0)) CATALOG(M.CEINS0)
```

5.1.4 Step 4 Kopieren der System-Datasets

Nun werden die Systemdateien vom originären / initialen System mit dem IEBCOPY-Utility auf die neue Systemresidenz REINS0 kopiert.
Dazu gehören das eigentliche System, der Nucleus (`<initial HLQ>.SYS1.NUCLEUS`), die Systemparameter, die in der Parameter-Bibliothek (`<initial HLQ>.SYS1.PARMLIB`) gespeichert sind, und die Procedure-Library (`<initial HLQ>.SYS1.PROCLIB`), die die Standard-Anwendungen beinhaltet.
Parmlib-Member ggf. auch Proclib-Member sind häufig mit sogenannten Suffixes „getauft“, die Rückschlüsse auf die Systeme, für die sie geplant sind, erlauben. Das erleichtert ggf. die weitere Administration.

Da die Datasets noch mit dem High-Level-Qualifier aus dem Ursprungssystem ausgestattet sind und dieser verhindert, dass das System überhaupt starten kann, wird der erste Qualifier entfernt, so dass die Dataset-Namen jetzt mit dem High-Level-Qualifier **‘SYS1‘** beginnen.

Listing 5.4: JOB004 Step 1 Kopieren SYSRES-Datasets

```
//COPY1    EXEC PGM=IEBCOPY,REGION=6M
//*******************************************************************
//*         COPY SYSRES DATASETS                                    *
//*******************************************************************
//SYSPRINT  DD SYSOUT=*
//IN1       DD DISP=SHR,DSN=ULISTOR.SYS1.SYSEINS.LOAD
//OUT1      DD UNIT=3390,VOL=SER=REINS0,DSN=ULISTOR.SYS1.SYSEINS.LOAD,
//             DISP=(,KEEP,DELETE),STORCLAS=NONSMS,
//             LIKE=ULISTOR.SYS1.SYSEINS.LOAD,
//             DCB=(RECFM=U,BLKSIZE=32760)
//SYSIN DD *
 COPY INDD=IN1,OUTDD=OUT1
```

Listing 5.5: JOB004 Step 2 Umbenennen Datasets

```
//*-------------------- IEHPROGM - RENAME ---------------------
//RENAME2 EXEC PGM=IEHPROGM
//SYSPRINT  DD SYSOUT=*
//ENQ1      DD UNIT=3390,VOL=SER=REINS0,DISP=OLD
//SYSIN     DD *
RENAME DSNAME=ULISTOR.SYS1.SYSEINS.LOAD,VOL=3390=REINS0,                +
               NEWNAME=SYS1.SYSEINS.LOAD
```

5.1.5 Step 5 Paging-Platten einrichten

Nach Anlage der Katalog-Datasets können auf der Paging-Platte die notwendigen Datenstrukturen für das Paging eingerichtet werden. Um das Paging effizient durchführen zu können, werden die Page-Bereiche in verschiedene Klassen eingeteilt:

- PLPA (Pageable Link Pack Area)
- COMMON
- LOCAL

Damit keine Konkurrenzsituation auftreten kann, werden die verschiedenen Page-Spaces normalerweise auf voneinander unabhängigen Platten abgelegt.
Das LOCAL-Page-Dataset kann auf mehrere Platten verteilt werden, um die Performance zu steigern. Das COMMON- und das PLPA-Page-Dataset (PLPA=Pageable Link Pack Area) sind ebenso wie das LOCAL-Page-Dataset notwendige Elemente für das Paging, dürfen aber im Gegensatz dazu nur einmal in einem System verwendet werden [sABC1]. Da es sich bei den hier zu Grunde liegenden Systemen jedoch um solche aus dem Lehrbereich mit relativ geringen Leistungsanforderungen und begrenzten Kapazitäten handelt, wurde an dieser Stelle darauf verzichtet und die drei Page-Spaces auf einer einzigen Platte abgelegt.

Listing 5.6: JOB005 Step 1 Aufbau Page-Spaces

```
//PAGEDS1 EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//PEINS0    DD UNIT=3390,VOL=SER=PEINS0,DISP=SHR
//SYSIN     DD *
  DEFINE PAGESPACE (NAME(EINSM.SYS1.EINS.PAGE.PLPA) FILE(PEINS0) -
                    CYLINDERS(1)    VOLUME(PEINS0) STORCLAS(NONSMS) UNIQUE)
  DEFINE PAGESPACE (NAME(EINSM.SYS1.EINS.PAGE.COMMON) FILE(PEINS0) -
                    CYLINDERS(299) VOLUME(PEINS0) STORCLAS(NONSMS) UNIQUE)
  DEFINE PAGESPACE (NAME(EINSM.SYS1.EINS.PAGE.LOCAL1) FILE(PEINS0) -
                    CYLINDERS(400) VOLUME(PEINS0) STORCLAS(NONSMS) UNIQUE)
 ALTER EINSM.SYS1.EINS.PAGE.PLPA          NEWNAME(SYS1.EINS.PAGE.PLPA)
 ALTER EINSM.SYS1.EINS.PAGE.PLPA.DATA     NEWNAME(SYS1.EINS.PAGE.PLPA.DATA)
 ALTER EINSM.SYS1.EINS.PAGE.COMMON        NEWNAME(SYS1.EINS.PAGE.COMMON)
 ALTER EINSM.SYS1.EINS.PAGE.COMMON.DATA NEWNAME(SYS1.EINS.PAGE.COMMON.DATA)
 ALTER EINSM.SYS1.EINS.PAGE.LOCAL1        NEWNAME(SYS1.EINS.PAGE.LOCAL1)
 ALTER EINSM.SYS1.EINS.PAGE.LOCAL1.DATA NEWNAME(SYS1.EINS.PAGE.LOCAL1.DATA)
```

Staging-Index-Datasets werden ebenfalls hier definiert, haben aber inzwischen eine eher marginale Bedeutung im Zusammenhang mit Datasets im sogenannten VIO-Bereich (siehe weiter unten SMS-Implementierung unter 5.1.5 auf Seite 54).

Listing 5.7: JOB005 Step 2 Aufbau STGINDEX

```
//STGIX2   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//PEINS0    DD UNIT=3390,VOL=SER=PEINS0,DISP=SHR
//SYSIN     DD *
  DEFINE CLUSTER (NAME(SYS1.EINS.STGINDEX) FILE(PEINS0) -
                  KEYS(12,8) CYLINDERS(2) RECORDSIZE(2041,2041) REUSE -
                  VOLUME(PEINS0) STORCLAS(NONSMS) BUFFERSPACE(20480) ) -
  DATA           (CONTROLINTERVALSIZE(2048)) -
  INDEX          (CONTROLINTERVALSIZE(1024)) -
  CATALOG        (M.CEINS0)
```

Bedeutsamer sind dagegen die System-Management-Facility-Dateien (SMF-Datasets), in denen u.a. statistische Informationen über Jobs, Started-Tasks, TSO-User, Transaktionen etc. festgehalten werden. Solche Daten werden bei Problem- und Performance-Analysen, vor allem aber zum Accounting (Abrechnung von Jobs, Transaktionen etc. nach Verbrauch) und nicht zuletzt von den Monitoring-Werkzeugen vielfältig ausgewertet [sABC1].

Listing 5.8: JOB005 Step 3 Aufbau SMF-Datasets

```
//SMFMAN3 EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//PEINS0    DD UNIT=3390,VOL=SER=PEINS0,DISP=SHR
//SYSIN     DD *
  DEFINE CLUSTER (NAME(SYS1.EINS.MAN1) FILE(PEINS0) -
                  VOLUME(PEINS0) STORCLAS(NONSMS) CYLINDERS(30)  -
                  RECORDSIZE(4086,32767) CONTROLINTERVALSIZE(4096) -
                  SHAREOPTIONS(2) NONINDEXED SPANNED REUSE SPEED) -
  CATALOG        (M.CEINS0)
  DEFINE CLUSTER (NAME(SYS1.EINS.MAN2) FILE(PEINS0) -
                  VOLUME(PEINS0) STORCLAS(NONSMS) CYLINDERS(10) -
                  RECORDSIZE(4086,32767) CONTROLINTERVALSIZE(4096) -
                  SHAREOPTIONS(2) NONINDEXED SPANNED REUSE SPEED) -
  CATALOG        (M.CEINS0)
  DEFINE CLUSTER (NAME(SYS1.EINS.MAN3) FILE(PEINS0) -
                  VOLUME(PEINS0) STORCLAS(NONSMS) CYLINDERS(10) -
                  RECORDSIZE(4086,32767) CONTROLINTERVALSIZE(4096) -
                  SHAREOPTIONS(2) NONINDEXED SPANNED REUSE SPEED) -
  CATALOG        (M.CEINS0)
```

5.1.6 Step 6 SMS (System Managed Storage) einrichten

Jetzt werden die Dateien für SMS (Subsystem für die Funktion System-Managed-Storage) eingerichtet. Dazu werden die Dateien Source-Control-Data-Set (SCDS), Active-Control-Data-Set (ACDS) und Communications-Data-Set (COMMDS) erzeugt.

Listing 5.9: JOB006 Step 1 Anlegen SMS-Datasets

```
//SMS1     EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//CEINS0    DD UNIT=3390,VOL=SER=CEINS0,DISP=SHR
//SYSIN     DD *
  DEFINE CLUSTER ( NAME( EINSU.SMS.STORPLEX.COMMDS ) -
                FILE(CEINS0) VOLUME(CEINS0) STORCLAS(NONSMS) TRACKS(1,1) -
                SHAREOPTIONS(3,3) LINEAR) -
         DATA( NAME( EINSU.SMS.STORPLEX.COMMDS.DATA ) )
  DEFINE CLUSTER ( NAME( EINSU.SMS.STORPLEX.ACDS )    -
                FILE(CEINS0) VOLUME(CEINS0) STORCLAS(NONSMS) -
                TRACKS(1,1) SHAREOPTIONS(3,3) LINEAR) -
         DATA( NAME( EINSU.SMS.STORPLEX.ACDS.DATA ) )
  DEFINE CLUSTER ( NAME( EINSU.SMS.STORPLEX.SCDS )   -
                FILE(CEINS0) VOLUME(CEINS0) STORCLAS(NONSMS) TRACKS(6,6) -
                SHAREOPTIONS(3,3) LINEAR) -
         DATA( NAME( EINSU.SMS.STORPLEX.SCDS.DATA ) )
```

Listing 5.10: JOB006 Step 2 Reproduktion SCDS

```
//REPRO1   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  REPRO   IDS(SYS1.SCDS) ODS(EINSU.SMS.STORPLEX.SCDS)
```

Listing 5.11: JOB006 Step 3 Umbenennen

```
//RENSYS   EXEC ACBJBAOB
//STEP2.SYSUDUMP DD  SYSOUT=*
//STEP2.SYSTSIN  DD *
PROFILE NOPREFIX
ISPSTART CMD(ACBQBAB1 ALTER SCDS('EINSU.SMS.STORPLEX.SCDS') +
               RENSYS(ADCD,EINS) ) +
               BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(999999)
ISPSTART CMD(ACBQBAB1 ALTER SCDS('EINSU.SMS.STORPLEX.SCDS') +
               ADDSYS(ZWEI) ) +
               BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(999999)
```

Listing 5.12: JOB006 Step 4 Umbenennen/Entfernen HLQ

```
//RENAME   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  ALTER EINSU.SMS.STORPLEX.SCDS          NEWNAME(SMS.STORPLEX.SCDS) -
        CATALOG(U.DEINS0)
  ALTER EINSU.SMS.STORPLEX.SCDS.DATA     NEWNAME(SMS.STORPLEX.SCDS.DATA) -
        CATALOG(U.DEINS0)
  ALTER EINSU.SMS.STORPLEX.ACDS          NEWNAME(SMS.STORPLEX.ACDS) -
        CATALOG(U.DEINS0)
  ALTER EINSU.SMS.STORPLEX.ACDS.DATA     NEWNAME(SMS.STORPLEX.ACDS.DATA) -
        CATALOG(U.DEINS0)
  ALTER EINSU.SMS.STORPLEX.COMMDS        NEWNAME(SMS.STORPLEX.COMMDS) -
        CATALOG(U.DEINS0)
  ALTER EINSU.SMS.STORPLEX.COMMDS.DATA NEWNAME(SMS.STORPLEX.COMMDS.DATA) -
        CATALOG(U.DEINS0)
```

SMS übernimmt viele Aufgaben, die früher manuell erledigt werden mussten und automatisiert diese. Es ist Teil des „Data-Facility-Storage-Management-Subsystem“ (DFSMS). Es trennt die logische von der physischen Perspektive.
Entwickler, Tester oder Arbeitsvorbereiter haben eine eher logische Sicht auf die Daten bzw. Dateien. Für sie ist es normalerweise wenig oder gar nicht bedeutsam, wo Daten physisch gespeichert werden. Dagegen haben System-Verantwortliche oder Personen, die sich mit Speichermanagement, Backup-Recovery-Betrachtungen, Disaster-Vermeidung oder Performance-Management befassen müssen, eine mehr physische Sicht auf die Daten respektive Dateien. Für diese Aufgaben ist es bedeutsam, wo bzw. auf welchen physischen Datenträgern Dateien sich befinden, um der entsprechenden Aufgabenstellung gerecht zu werden [zABC3].

SMS stellt verschiedene Konzepte zur Verfügung, die die Trennung von Logik und Physik ermöglichen:

- **Daten-Klasse / Data-Class**
  Daten-Klassen lassen sich sowohl auf SMS-verwaltete als auch auf nicht-SMS-verwaltete Datasets übertragen, nicht aber auf Objekte.
  Durch diese Klassen können Allocationen von Datasets vereinfacht werden, indem statt einer langen Kette von Parametern oder Schlüsselwörtern eine Daten-Klasse angegeben wird. In einer Daten-Klasse können Informationen über den benötigten Speicher, VSAM-Attribute, Volumen-Zähler oder Tape-Attribute zusammengefasst werden. Diese Informationen werden so zu Standard-Werten (Default-Werte), die

immer dann gelten, wenn nichts anderes bestimmt ist. Viele dieser Werte können duch explizite Angabe z.B. in den Job-Steuer-Anweisungen (JCL - Job Control Language) überschrieben werden.

- **Speicher-Klassen / Storage-Class**
  Speicher-Klassen lassen sich auf SMS-verwaltete Datasets und Objekte anwenden. Dadurch werden unter anderem verschiedene Service- und Verfügbarkeits-Levels definiert bzw. bereitgestellt. Speicherklassen separieren den von den Objekten oder Datasets benötigten Service von den physischen Eigenschaften. Die Verbindung von einer Speicher-Klasse mit einem Objekt oder einem Dataset machen die Verwaltung durch SMS notwendig.

- **Management-Klasse / Management-Class**
  Diese Klasse lässt sich auf Direct-Access-Storage-Device-Datasets (DASD-Datasets) und Objekte anwenden. Management-Klassen enthalten Migrations-, Backup- und Selbsterhaltungs-Informationen für DASD-Datasets sowie Backup-Anforderungen und Übergangskriterien für Objekte. Die Attribute aus der Management-Klasse stellen Teile für die Speicherverwaltung und Verfügbarkeitsrichtlinien zur Verfügung. Des Weiteren stellen sie Migrations-, Expirations- und Backup-Informationen bereit.
  Ein Objekt muss mit einer Management-Klasse assoziiert sein.

- **Speichergruppen / Storage-Group**
  Speichergruppen lassen sich nur auf SMS-verwaltete Datasets und Objekte anwenden. Sie repräsentieren die physischen Speicher, auf denen die Daten gespeichert sind. Das Gruppieren von Platten oder Pools kann dynamisch modifiziert werden, ohne dass dafür ein „Initial Program Load“ (IPL) nötig wäre.
  Es gibt sechs Typen von Speichergruppen:
  - **Pool**: Der Pool enthält die Namen aller von ihm zu verwaltenden DASD-Volumes.
  - **Dummy**: Die Dummygruppe beinhaltet die Volume-Serials von Volumes, die z.B. augenblicklich nicht mehr oder noch nicht mit dem System verbunden sind.
  - **Virtual Input/Output (VIO)**: Diese Gruppe enthält keine Volumes. Sie allokiert im Pagingbereich des virtuellen Speichers Datasets und simuliert auf diese Weise ein DASD-Volume.

- **Objekt**: Diese Gruppe enthält optische Volumes und DASD-Volumes, die für Objekte benötigt werden.
- **Objekt Backup**: Enthält die DASD-Volumes, die für Backup-Kopien von Objekten benötigt werden.
- **Tape**: Enthält SMS-verwaltete Tapes

Alle von SMS verwalteten Datasets und Objekte müssen einer Speichergruppe zugeordnet sein. Dabei spielen Pool-Speichergruppen eine besondere Rolle, denn sie bieten die Möglichkeit mehrere DASD-Volumes als Gruppe anzusprechen, ohne sich auf ein bestimmtes Volume beziehen zu müssen.

Jedes SMS hat seine eigene Basiskonfiguration. In dieser Konfiguration ist enthalten, welche Systeme durch SMS verwaltet werden. In der Basiskonfiguration sind folgende Defaultwerte festgelegt:

- **Default Management-Klasse**: Name der Verwaltungsklasse, die benutzt werden soll, wenn für eine Dataset oder Objekt keine Verwaltungsklasse angegeben wurde.
- **Default Unit**: Der Name für nicht SMS-verwaltete Allokierungen, falls keine Unit angegeben wurde.
- **Default Laufwerks-Geometrie**: Anzahl der Bytes pro Track, Tracks pro Zylinder etc.

SMS vereinfacht das Verwalten von Daten auf Festplatten, indem es bei vielen Aufgaben z.B. beim Anlegen von Dateien auf bestimmten Festplatten, bei der Festlegung der Charakteristika von Datasets usw. steuernd eingreift. Hierzu werden sogenannte ACS-Routinen (Automated Control Storage) definiert, die nach vorgegebenen Regeln den Plattenspeicher verwalten. Datasets werden dabei mit Hilfe ihres Namens vorher festgelegten Plattenpools zugeordnet. Die Verwaltung von SMS kann über das „Interactive Storage Management Facility“ (ISMF) unter TSO erfolgen. Die SMS-Struktur muss im gesamten Parallel-Sysplex-Verbund einheitlich sein [zABC3].

5.1.7 Step 7 Input-Output-Definition-File (IODF) implementieren

Um externe Geräte ansprechen zu können, benötigt das Betriebssystem eine Definition der Eingabe- und Ausgabe-Devices. Diese Definitionen werden im ‘Input-Output-Definition-File‘ (IODF) gespeichert. Diese Datei wird über eine IODF-Anweisung beim IPL angezogen, so dass die Definitionen vor dem Systemstart bekannt sein müssen. Es

können mehrere IODFs für ein Betriebssystem existieren. Um diese unterscheiden zu könnenk, wird an den Namen SYS1.IODFxx ein Suffix aus zwei Zeichen angehängt. Dies ist die übliche Vorgehensweise, um in einem z/OS Betriebssystem Ketten (Konkatenierung) von Definitionen anzusprechen [sABC1].

Listing 5.13: JOB007 Aufbau IODF-Dataset

```
//DFIODF1 EXEC PGM=IDCAMS                    /* DEL/DEF EINSU.IODF00.CLUSTER */
//SYSPRINT  DD SYSOUT=*
//CEINS0    DD UNIT=3390,VOL=SER=CEINS0,DISP=SHR
//SYSIN     DD *
  DEL IODF.IODF00.CLUSTER CAT(U.DEINS0) FILE(CEINS0)
  SET MAXCC=00
  DEFINE CLUSTER(NAME(EINSU.IODF00.CLUSTER) LINEAR SHAREOPTIONS(1 3) -
                 UNIQUE STORCLAS(NONSMS) VOLUMES(CEINS0)) -
         DATA(NAME(EINSU.IODF00) RECORDS(512))
//*
//INIODF2 EXEC PGM=CBDMGHCP,                 /* INITIALIZE IODF00 CLUSTER */
//             PARM='INITIODF SIZE=512,ACTLOG=NO'
//HCDIODFT  DD DISP=SHR,DSN=EINSU.IODF00
//HCDMLOG   DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=6650)
//*
//CPIODF3 EXEC PGM=CBDMGHCP,      /* COPY SYS1.IODF00 TO EINSU.IODF00 */
//             PARM='COPYIODF'
//HCDIODFS  DD DISP=SHR,DSN=SYS1.IODF99
//HCDIODFT  DD DISP=SHR,DSN=EINSU.IODF00
//HCDMLOG   DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=6650)
//*
//ALTER4  EXEC PGM=IDCAMS       /* RENAME EINSU.IODF00 TO IODF.IODF00 */
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  ALTER  EINSU.IODF00.CLUSTER NEWNAME(IODF.IODF00.CLUSTER) -
         CATALOG(U.DEINS0)
  ALTER  EINSU.IODF00         NEWNAME(IODF.IODF00) -
         CATALOG(U.DEINS0)
```

5.1.8 Step 8 Resource Access Control Facility (RACF) einrichten

Um das Security-Umfeld von z/OS einzurichten, wird jetzt die RACF-Datenbank erzeugt.

Listing 5.14: JOB008 Teil 1 Aufbau RACF-Dataset

```
//DELNVS1 EXEC PGM=IDCAMS     /* DELETE SYS1.RACF + SYS1.RACF.BACKUP */
//CEINS0    DD UNIT=3390,VOL=SER=CEINS0,DISP=SHR
//SYSPRINT  DD SYSOUT=*
```

```
  DEL  SYS1.RACF          CAT(M.CEINS0)  FILE(CEINS0)
  DEL  SYS1.RACF.BACKUP CAT(M.CEINS0)  FILE(CEINS0)
//*
//COPY1    EXEC PGM=IRRUT400,PARM=NOLOCKINPUT      /* CREATE SYS1.RACF */
//SYSPRINT  DD SYSOUT=*
//INDD1     DD DISP=SHR,DSN=SYS1.RACFDS
//OUTDD1    DD DSN=SYS1.RACF,DISP=(NEW,KEEP),UNIT=3390,VOL=SER=CEINS0,
//             SPACE=(CYL,(5),,CONTIG),DCB=DSORG=PSU
//*
```

RACF steht für Resource Access Control Facility. Diese Einrichtung hat hauptsächlich folgende Funktionen:

- **Kontrolle des Systemzugriffs** über Benutzerkennung und Passwort
- **Einschränkung der Aktionen**, die ein Benutzer ausführen kann:
  - **mit Datasets** auf Grund von Dataset-Profilen mit den Rechten:
    - **None** = Keinerlei Rechte
    - **Read** = Lese-Zugriff
    - **Control** = Inhalt darf geändert werden (fast wie Update)
    - **Update** = Inhalt darf geändert werden
    - **Alter** = Änderung von Datei-Attributen incl. Löschen der Datei
  - **mit Programmen** (**Exec** = Ausführung erlaubt, **None** = keinerlei Rechte)
- **Flexible Kontrolle** über den Zugriff auf geschützte Ressourcen (Auditing)
- **Schutz der Installationsressourcen**
- **Möglichkeit Informationen** über andere Produkte zu speichern
- **Zentralisierte oder dezentralisierte Kontrolle** der Profile
- **ISPF-Interface** unter TSO (Terminal-Einrichtung unter z/OS)
  ISPF=Interactive System Productivity Facility = Benutzeroberfläche unter TSO [sABC1].

Aus Sicherheitsgründen wird die RACF-Datenbank in zweiter Ausfertigung im SYS1.RACF.BACKUP-Dataset angelegt.

Listing 5.15: JOB008 Teil 2 Aufbau RACF-BACKUP-Dataset

```
//COPY2    EXEC PGM=IRRUT400,PARM=NOLOCKINPUT   /* CREATE RACF.BACKUP */
//SYSPRINT  DD SYSOUT=*
//INDD1     DD DISP=SHR,DSN=SYS1.RACFDS.BACKUP
//OUTDD1    DD DSN=SYS1.RACF.BACKUP,DISP=(NEW,KEEP),UNIT=3390,
//             VOL=SER=CEINS0,SPACE=(CYL,(5),,CONTIG),DCB=DSORG=PSU
//*
//*
//DEFNVS1 EXEC PGM=IDCAMS      /* CATALOG SYS1.RACF + SYS1.RACF.BACKUP */
//CEINS0    DD UNIT=3390,VOL=SER=CEINS0,DISP=SHR
//SYSPRINT  DD SYSOUT=*
  DEFINE NONVSAM(NAME(SYS1.RACF) DEVT(3390) VOLUMES(CEINS0)) -
         CATALOG(M.CEINS0)
  DEFINE NONVSAM(NAME(SYS1.RACF.BACKUP) DEVT(3390) VOLUMES(CEINS0)) -
         CATALOG(M.CEINS0)
```

5.1.9 Step 9 Job Entry System (JES) installieren

Die Installation von JES2 (oder stattdessen JES3) bietet die Möglichkeit, einen Spoolbereich für alle Systeme im Sysplex gemeinsam zu nutzen. Diese Einrichtung nennt sich 'Multi-Access-Spool' (MAS)[1]. Eine kurze Beschreibung von 'MAS' wird unter 5.2 auf Seite 78 gegeben.

Listing 5.16: JOB009 Aufbau MAS-Datasets

```
//CKPSPL2 EXEC PGM=IEFBR14                          /***** ALLOCATE *****/
//HASPCK01 DD DISP=(,KEEP,DELETE),UNIT=3390,DSN=SYS1.STORPLEX.HASPCK01,
//            SPACE=(CYL,(2),,CONTIG),VOL=SER=PEINS0,DCB=DSORG=PSU
//HASPCK02 DD DISP=(,KEEP,DELETE),UNIT=3390,DSN=SYS1.STORPLEX.HASPCK02,
//            SPACE=(CYL,(2),,CONTIG),VOL=SER=PEINS0,DCB=DSORG=PSU
//HASPCKN1 DD DISP=(,KEEP,DELETE),UNIT=3390,DSN=SYS1.STORPLEX.HASPCKN1,
//            SPACE=(CYL,(2),,CONTIG),VOL=SER=PEINS0,DCB=DSORG=PSU
//HASPCKN2 DD DISP=(,KEEP,DELETE),UNIT=3390,DSN=SYS1.STORPLEX.HASPCKN2,
//            SPACE=(CYL,(2),,CONTIG),VOL=SER=PEINS0,DCB=DSORG=PSU
//HASPACE  DD DISP=(,KEEP,DELETE),UNIT=3390,DSN=SYS1.STORPLEX.HASPACE,
//            SPACE=(CYL,(200),,CONTIG),VOL=SER=PEINS0,DCB=DSORG=PSU
//HASPIDX  DD DISP=(,KEEP,DELETE),UNIT=3390,DSN=ISF.HASPINDX,
//            DCB=(LRECL=4096,BLKSIZE=4096,RECFM=F,DSORG=DA),
//            SPACE=(CYL,(1)),STORCLAS=NONSMS,VOL=SER=PEINS0
//*
//CATNVSAM EXEC PGM=IDCAMS                          /***** CATALOG *****/
```

[1] JES3 bietet ein ähnliches Konstrukt, das sich „COMPLEX“ nennt.

```
//SYSPRINT  DD SYSOUT=*
//PEINS0    DD UNIT=3390,VOL=SER=PEINS0,DISP=SHR
//SYSIN     DD *
  DEFINE NONVSAM (NAME(SYS1.STORPLEX.HASPCK01) DEVT(3390) VOL(PEINS0)) +
         CATALOG (M.CEINS0)
  DEFINE NONVSAM (NAME(SYS1.STORPLEX.HASPCK02) DEVT(3390) VOL(PEINS0)) +
         CATALOG (M.CEINS0)
  DEFINE NONVSAM (NAME(SYS1.STORPLEX.HASPCKN1) DEVT(3390) VOL(PEINS0)) +
         CATALOG (M.CEINS0)
  DEFINE NONVSAM (NAME(SYS1.STORPLEX.HASPCKN2) DEVT(3390) VOL(PEINS0)) +
         CATALOG (M.CEINS0)
  DEFINE NONVSAM (NAME(SYS1.STORPLEX.HASPACE ) DEVT(3390) VOL(PEINS0)) +
         CATALOG (M.CEINS0)
  DEFINE NONVSAM (NAME(ISF.HASPINDX          ) DEVT(3390) VOL(PEINS0)) +
         CATALOG (M.CEINS0)
```

JES ist zuständig für das Management von Jobs und den damit verbundenen Ein- und Ausgaben. Zu den weiteren Aufgaben zählen aber auch das Handling von sogenannten „Started Tasks“ (‘STC‘) und der „TSO-Anwender“(‘TSU‘).
Die Verarbeitung von Batch-Jobs wird durch die „Initiators“ kontrolliert. Dabei werden sowohl Klassen, Prioritäten als auch die Anzahl der Jobs, die parallel auf dem System arbeiten dürfen, definiert. Die Eingabe- und Ausgabe-Streams werden in einem zentralen Spool-Dataset gespeichert. JES arbeitet mit sog. „Checkpoints“. Die Checkpoints sollen verhindern, dass bei gravierenden Fehlern das System zum vollständigen Stillstand kommt und dass in diesem Fall möglichst wenige Daten verloren gehen. Um dieses Konzept zu verwirklichen, gibt es neben dem „Checkpointing“ auch noch den „Checkpoint Reconfiguration Dialog“, den man der System-Console ausführen kann.

Dazu schreibt JES in regelmäßigen Abständen die veränderten Daten in die Checkpoint-Datasets, während die Ursprungsdaten erst verändert werden, wenn der Job erfolgreich zu einem Ende gekommen ist. Sollte nun im Laufe des Jobs ein Fehler auftreten, der den Job oder das ganze System zum Absturz bringt, ist der Datenverlust minimal, da man mit Hilfe dieser Checkpoints wieder zu einem konsistenten Zustand der Daten zurückkehren kann [zABC2].

5.1.10 Step 10 BROADCAST- und LOGREC-Datasets einrichten

Nach der JES-Installation werden jetzt die BROADCAST- und die LOGREC-Datasets allokiert. Das Broadcast-Dataset (systemintern auch BRODCAST[2] genannt) wird weiter

[2] wegen der Benutzung als Teil des Dateinamens wird der Begriff auf maximal 8 Zeichen beschränkt

unten unter 5.1.17 auf Seite 69 näher erklärt.

Listing 5.17: JOB010 Aufbau LOGREC- and BROADCAST-Datasets

```
//*                          INIT  EINSM.SYS1.EINS.LOGREC + EINSM.SYS1.DAESHARE
//LOGREC1 EXEC PGM=IFCDIP00
//SERERDS  DD DSN=EINSM.SYS1.EINS.LOGREC,DISP=(NEW,CATLG),UNIT=3390,
//            VOLUME=(,RETAIN,SER=PEINS0),STORCLAS=NONSMS,
//            SPACE=(CYL,5)
//DAE      DD DSN=EINSM.SYS1.DAESHARE,DISP=(NEW,CATLG),
//            SPACE=(CYL,(1)),STORCLAS=NONSMS,VOL=SER=PEINS0,UNIT=3390,
//            DCB=(LRECL=255,RECFM=FB,BLKSIZE=23205)
//*
//BRODCS2 EXEC PGM=IEFBR14          /* ALLOC EINSM.SYS1.EINS.BROADCAST */
//SYSUT2   DD DISP=(,CATLG),UNIT=3390,VOL=SER=PEINS0,SPACE=(CYL,1),
//            STORCLAS=NONSMS,DCB=(LRECL=00129,BLKSIZE=00129,RECFM=F),
//            DSN=EINSM.SYS1.EINS.BRODCAST
//*
//BRODCS4 EXEC PGM=IDCAMS       /* ALTER DS-NAMES IN CATALOG M.CEINS0 */
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 ALTER EINSM.SYS1.EINS.BRODCAST -
             NEWNAME(SYS1.EINS.BRODCAST) CAT(M.CEINS0)
 ALTER EINSM.SYS1.EINS.LOGREC -
             NEWNAME(SYS1.EINS.LOGREC) CAT(M.CEINS0)
 ALTER EINSM.SYS1.DAESHARE    -
             NEWNAME(SYS1.DAESHARE) CAT(M.CEINS0)
```

Im LOGREC-Dataset werden Logging-Daten gespeichert. Darin werden sowohl statistische Informationen, Maschinen- und Programm-Fehler als auch Informationen über bestimmte „Interrupts“ und über dynamische Laufwerks-Rekonfiguration(en) aufgezeichnet.

Insbesondere bei Fehlern im System ist das LOGREC-Dataset ein nützliches Hilfsmittel, um die Ursache des Fehlers zu finden. Das System fertigt bei einem Fehler einen „Snapshot“ von sich selbst an, in dem es alle relevanten Systeminformationen zum Zeitpunkt des Auftretens des Fehlers in das Logrec-Dataset schreibt [SPGLO].

5.1.11 Step 11 Datasets für Virtual-Telecommunication-Acccess-Method (VTAM) erzeugen

VTAM ist die proprietäre IBM-Telekommunikations-Methode für ein IBM-Netzwerk, welches dem IBM-Konzept der System-Network-Architecture (SNA) entspricht. Mehrere solcher SNA-Netze werden über das Advanced-Peer-to-Peer-Networking-Verfahren

(kurz APPN) miteinander vernetzt (entspricht in etwa dem IP-Protokoll bei TCP/IP) [zABC7].
Die für VTAM/APPN benötigten Informationen zum Betrieb und zur Beschreibung der Topologie werden in den entsprechenden VTAM- bzw. APPN-Dateien gespeichert. Diese werden jetzt für den Master-Klon installiert. Änderungen an der Topologie oder an den Betriebsparametern werden in diesen Dateien vorgenommen und können meistens durch sogenannte Operator-Kommandos im laufenden Betrieb aktiviert werden.

Folgenden 4 Dateien werden mit Aufruf des Programms IEFBR14 erzeugt:

- SYS1.EINS.DSDBCTRL (Directory Service Database Control)
- SYS1.EINS.DSDB1 (Directory Services Database 1)
- SYS1.EINS.DSDB2 (Directory Services Database 2)
- SYS1.EINS.TRSDB

Listing 5.18: JOB011 Anlegen VTAM/APPN-Datasets

```
//ALLOCATE  EXEC PGM=IEFBR14
//DD1       DD DISP=(,KEEP),DSN=SYS1.EINS.DSDBCTRL,UNIT=3390,
//            VOL=SER=PEINS0,SPACE=(CYL,(1,1)),STORCLAS=NONSMS,
//            DCB=(LRECL=20,BLKSIZE=20,RECFM=F)
//DD2       DD DISP=(,KEEP),DSN=SYS1.EINS.DSDB1,UNIT=3390,
//            VOL=SER=PEINS0,SPACE=(CYL,(1,1)),STORCLAS=NONSMS,
//            DCB=(LRECL=1000,BLKSIZE=27000,RECFM=FB)
//DD3       DD DISP=(,KEEP),DSN=SYS1.EINS.DSDB2,UNIT=3390,
//            VOL=SER=PEINS0,SPACE=(CYL,(1,1)),STORCLAS=NONSMS,
//            DCB=(LRECL=1000,BLKSIZE=27000,RECFM=FB)
//DD4       DD DISP=(,KEEP),DSN=SYS1.EINS.TRSDB,UNIT=3390,
//            VOL=SER=PEINS0,SPACE=(CYL,(1,1)),STORCLAS=NONSMS,
//            DCB=(LRECL=1000,BLKSIZE=27000,RECFM=FB)
```

Dann werden sie als Non-VSAM-Dateien im Master-Katalog mit Hilfe des Programms IDCAMS (VSAM-Utility-Programm) eingetragen:

```
DEF NVSAM (NAME(<Name des Datasets>) VOL(PZWEI0) DEVT(3390))CAT(M.CEINS0)
```

Listing 5.19: JOB011 Katalogisieren VTAM/APPN-Datasets

```
//CATALOG   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
```

```
  DEF NVSAM (NAME(SYS1.EINS.DSDBCTRL) VOL(PEINS0) DEVT(3390)) -
             CAT(M.CEINS0)
  DEF NVSAM (NAME(SYS1.EINS.DSDB1) VOL(PEINS0) DEVT(3390)) -
             CAT(M.CEINS0)
  DEF NVSAM (NAME(SYS1.EINS.DSDB2) VOL(PEINS0) DEVT(3390)) -
             CAT(M.CEINS0)
  DEF NVSAM (NAME(SYS1.EINS.TRSDB) VOL(PEINS0) DEVT(3390)) -
             CAT(M.CEINS0)
```

5.1.12 Step 12 Registrieren der Non-VSAM-Datasets im Master-Katalog

Nun müssen noch die sogenannten Non-VSAM-Datasets der System-Residenz „REINS0“ mit IDCAMS im Master-Katalog eingetragen werden.

Listing 5.20: JOB012 Katalogisieren NVSAM-Datasets

```
//CATLG1   EXEC PGM=IDCAMS
//*******************************************************************
//*          CATALOG REINS0 DATASETS                                *
//*******************************************************************
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  DEF NVSAM(NAME(SYS1.SYSEINS.LOAD     ) VOL(&SYSR1) DEVT(0000)) +
      CAT(M.CEINS0)
  DEF NVSAM(NAME(SYS1.NUCLEUS          ) VOL(&SYSR1) DEVT(0000)) +
      CAT(M.CEINS0)
  DEF NVSAM(NAME(SYS1.PARMLIB          ) VOL(******) DEVT(0000)) +
      CAT(M.CEINS0)
  DEF NVSAM(NAME(SYS1.PROCLIB          ) VOL(&SYSR1) DEVT(0000)) +
      CAT(M.CEINS0)
  DEF NVSAM(NAME(SYS1.SVCLIB           ) VOL(&SYSR1) DEVT(0000)) +
      CAT(M.CEINS0)
  DEF NVSAM(NAME(SYS1.VTAMLST          ) VOL(&SYSR1) DEVT(0000)) +
      CAT(M.CEINS0)
```

5.1.13 Step 13 Vorbereiten ‘JOB014‘ zum Katalogisieren der übrigen Non-VSAM-Dateien

In diesem Schritt wird nun das Job-Member ‘JOB014‘ erzeugt, in dem zunächst eine temporäre Liste (`&&VTOC`) mit dem Datenträgerinhaltsverzeichnis (VTOC) der Referenz-Platten `Z8RES1, Z8RES2, Z8RES3 und Z8USS1` erstellt wird. Anschließend wird diese

temporäre Liste in einer Prozedur namens RCATDS eingelesen und daraus dann der JOB014 erzeugt - siehe: ‘`//CATALOG DD DSN=ULISTOR.SYSPLEX.CNTL(JOB014),DISP=SHR`‘.

Sofern man dieses Klon-Verfahren auf einen anderen Rechner-Komplex überträgt, muss dieser Job hier speziell angepasst werden und auf die Plattennamen des anderen Sysplex' verweisen.

Listing 5.21: JOB013 Vorbereiten JOB014

```
//S1  EXEC PGM=IEHLIST
//SYSPRINT DD DISP=(,PASS),UNIT=SYSDA,SPACE=(CYL,(10,2),RLSE),DSN=&&VTOC
//Z8RES1   DD DISP=SHR,UNIT=3390,VOL=SER=Z8RES1
//Z8RES2   DD DISP=SHR,UNIT=3390,VOL=SER=Z8RES2
//Z8RES3   DD DISP=SHR,UNIT=3390,VOL=SER=Z8RES3
//Z8USS1   DD DISP=SHR,UNIT=3390,VOL=SER=Z8USS1
//SYSIN    DD *
 LISTVTOC VOL=3390=Z8RES1
 LISTVTOC VOL=3390=Z8RES2
 LISTVTOC VOL=3390=Z8RES3
 LISTVTOC VOL=3390=Z8USS1
//S2      EXEC PGM=IKJEFT01
//SYSEXEC  DD DSN=ULISTOR.SYSPLEX.CNTL,DISP=SHR
//LISTVTOC DD DSN=&&VTOC,DISP=(OLD,DELETE)
//CATALOG  DD DSN=ULISTOR.SYSPLEX.CNTL(JOB014),DISP=SHR
//SYSTSPRT  DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SYSTSIN   DD *
 PROFILE NOPREFIX
 RCATDS EINS 0
```

5.1.14 Step 14 Katalogisieren der in Step 13 ermittelten Non-VSAM-Dateien

Der in Step 13 erzeugte Job wird ausgeführt, um den Master-Katalog des Klons zu aktualisieren. Wichtig ist dabei, dass alle die hier angesprochenen Dateien auf symbolische Volumes (`&SYSR1, &SYSR2, &SYSR3, &SYSR4, &SYSR5`) verweisen. Bei IPL sind diese System-Symbole u.a. im Member IEASYMSY bestimmt worden. Im laufenden System gelten dann die folgenden Zuweisungen:

```
&SYSR1=REINS0

&SYSR2=Z8RES1

&SYSR3=Z8RES2
```

`&SYSR4=Z8RES3`

`&SYSR5=Z8USS1`

Wie man sehen kann, entsprechen die Symbole `&SYSR2-5` genau den Volumes aus Step 13. Auf diese Weise wird auf die existierende Infrastruktur eines anderen Systems auf den Platten `Z8RES1-3` und `Z8USS1` im Bedarfsfall zugegriffen.

Listing 5.22: JOB014 Katalogisieren Rest der Infrastruktur

```
//CATLG   EXEC  PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN     DD *
 DEF NVSAM(NAME(ADCD.Z18.CLIST)      DEVT(0000) VOL(&SYSR2))
...
 DEF NVSAM(NAME(MVS.ZOSR08.DOCLIB) DEVT(0000) VOL(&SYSR2))
 DEF NVSAM(NAME(SYS1.LINKLIB)       DEVT(0000) VOL(&SYSR2))
...
 DEF NVSAM(NAME(SYS1.LPALIB)        DEVT(0000) VOL(&SYSR2))
 DEF NVSAM(NAME(SYS1.MACLIB)        DEVT(0000) VOL(&SYSR2))
 DEF NVSAM(NAME(SYS1.VTAMLIB)       DEVT(0000) VOL(&SYSR2))
 DEF NVSAM(NAME(CSQ600.SCSQLOAD)    DEVT(0000) VOL(&SYSR3))
 DEF NVSAM(NAME(SYS1.Z17.SIATTBL0) DEVT(0000) VOL(&SYSR5))
...
```

5.1.15 Step 15 Schreiben des IPL-Bootstraps

In diesem Schritt wird der IPL-Text (Initial Program Load) vom Programm ICKDSF geschrieben. Als Eingabe dient die Datei-Verkettung

Listing 5.23: Konkatenierung der IPL-Bootstrap-Eingabe

```
//IPLTEXT   DD DISP=SHR,DSN=SYS1.SAMPLIB(IPLRECS)
//          DD DISP=SHR,DSN=SYS1.SAMPLIB(IEAIPL00)
```

Die Eingabe bildet das eigentliche Stückchen Code des Boot-Loaders und wird von ICKDSF auf die Platte REINS0 geschrieben. Dies entspricht in etwa dem DOS-Command `‘format a: /system‘`, bei dem auch das Ladeprogramm an eine feste Stelle auf die Disk geschrieben wird.

Listing 5.24: JOB015 Schreibe IPL-Bootstrap

```
//IPLTEXT EXEC PGM=ICKDSF
```

```
//SYSPRINT  DD SYSOUT=*
//IPLTEXT   DD DISP=SHR,DSN=SYS1.SAMPLIB(IPLRECS)
//           DD DISP=SHR,DSN=SYS1.SAMPLIB(IEAIPL00)
//IPLVOL    DD UNIT=3390,VOL=SER=REINS0,DISP=OLD
//SYSIN     DD *
 REFORMAT                    -
       DDNAME(IPLVOL)        -
       IPLDD(IPLTEXT,OBJ)    -
       PURGE                 -
       NOVERIFY              -
       OWNERID(IBM)          -
       BOOTSTRAP
```

5.1.16 Step 16 Erzeugen des Members SYSCATLG in SYS1.NUCLEUS

Mit dem Programm `IEHPROGM` wird das Member `SYSCATLG` aus der Datei `SYS1.NUCLEUS` entfernt und anschliessend mit Programm `IEBDG` mit den spezifischen Daten für dieses System neu aufgebaut. Auf die einzelnen Anweisungen für `IEBDG` wird nicht näher eingegangen. Das Ergebnis des Jobs ist ein Member `SYSCATLG`, welches nur einen einzelnen Datensatz beinhaltet. Dieser zeigt u.a. auf den Namen der Master-Katalog-Platte 'CEINS0' und auf den Master-Katalog-Namen 'M.CEINS0'. Das ganze wird so in den Installationsanweisungen vorgeschrieben.
In dem Buch IBM Redbook 'Introduction to the New Mainframe z/OS Basics' [IBM 1] findet man eine Beschreibung des IEBDG-Programms.

Listing 5.25: JOB016 Löschen+Aufbau SYSCATLG-Member

```
//SCRTCH1 EXEC PGM=IEHPROGM
//*        SCRATCH OLD SYSCATLG MEMBER                                *
//SYSPRINT  DD SYSOUT=*
//ENQ1      DD UNIT=3390,VOL=SER=REINS0,DISP=OLD
//SYSIN     DD *
  SCRATCH DSNAME=SYS1.NUCLEUS,VOL=3390=REINS0,MEMBER=SYSCATLG
//CATPTR2 EXEC PGM=IEBDG
//*        REPLACE SYSCATLG                                           *
//SYSPRINT  DD SYSOUT=*
//NUCLEUS   DD UNIT=3390,VOL=SER=REINS0,DISP=SHR,
//             DSN=SYS1.NUCLEUS(SYSCATLG)
//SYSIN     DD *
  DSD OUTPUT=(NUCLEUS)
  FD NAME=VOL,LENGTH=6,STARTLOC=01,FILL=X'40',PICTURE=6,'REINS0'
  FD NAME=CATTYPE,LENGTH=1,STARTLOC=07,FILL=X'F1'
  FD NAME=ALIAS,LENGTH=1,STARTLOC=08,FILL=X'F1'
```

```
  FD NAME=CASLL,LENGTH=2,STARTLOC=09,FILL=X'40'
  FD NAME=CAT,LENGTH=44,STARTLOC=11,FILL=X'40',PICTURE=8,'M.CEINS0'
  FD NAME=FIL,LENGTH=62,STARTLOC=19,FILL=X'40'
  CREATE QUANTITY=1,FILL=X'00',NAME=(VOL,CATTYPE,ALIAS,CASLL,CAT,FIL)
//* ----+----1----+----2----+----3----+----4
//* CEINS011   M.CEINS0
//* ------.-..--------
//*      |   || |      |---->  MASTERCATALOG NAME
//*      |   || +------------> CAS TASKS LOW LIMIT (DEFAULT 3C)
//*      |   |+-------------->  ALIAS LEVEL
//*      |   +--------------->  ICF CATALOG WITH SYS\% OFF
//*      +------------------->  MASTERCATALOG VOLSER
```

5.1.17 Step 17 Erstellen User-Attribut-Dataset (UADS) und BROADCAST-Dataset

In diesem Schritt werden das UADS- (User-Attribute-Dataset) und das Broadcast-Dataset installiert.
Das UADS beinhaltet eine Liste der Benutzer, die berechtigt sind, TSO zu nutzen, und wie diese auf das System zugreifen können. Für jeden Benutzer existiert ein Eintrag im UADS. Dabei werden die folgenden Informationen gespeichert:

- **User-ID**
- **Ein oder mehrere Passwörter** oder einen Null-Pointer, die der User-ID zugeordnet sind.
- **Eine oder mehrere Zugangs-Nummern** oder einen Null-Pointer, die den Passwörtern zugeordnet sind.
- **Ein oder mehrere Prozedurnamen**, die den Zugangs-Nummern zugeordnet sind: Jeder Prozedurname steht für eine Prozedur, die beispielsweise beim Anmelden an das System gestartet wird.
- **weitere Informationen** [UADS]

Listing 5.26: JOB017 Teil 1 Aufbau UADS

```
//*-------------------------------------------------------------------
//DEFNVS3 EXEC PGM=IDCAMS
//*                                                  CATALOG UADS
//CEINS0     DD UNIT=3390,VOL=SER=CEINS0,DISP=SHR
```

```
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  DELETE SYS1.EINS.UADS CATALOG(M.CEINS0) FILE(CEINS0)
  SET MAXCC=0
//COPYUA1 EXEC PGM=IEBCOPY,REGION=6M
//*                                          COPY SYS1.UADS TO CEINS0
//SYSPRINT  DD SYSOUT=*
//IN        DD DSN=SYS1.UADS,DISP=SHR
//OUT       DD DSN=SYS1.EINS.UADS,UNIT=3390,VOL=SER=CEINS0,
//            DISP=(,KEEP,DELETE),SPACE=(CYL,(1,1,20)),
//            STORCLAS=NONSMS,DCB=(LRECL=80,RECFM=FB,BLKSIZE=800)
//SYSIN     DD *
 COPY INDD=IN,OUTDD=OUT
```

Das Broadcast-Dataset dient der Aufnahme von Nachrichten, die an Benutzer von Terminals oder an solche Benutzergruppen addressiert sind. Darum wird das Broadcast-Dataset in Nachrichten für einzelne Benutzer oder für Benutzergruppen gesplittet.

Da sich die Benutzer im gesamten Sysplex anmelden können sollen, ist das UADS auf den gemeinsam genutzten Platten als globale Ressource angesiedelt.

Auch das Broadcast-Dataset kann als lokale oder globale Ressource genutzt werden.
Wegen der einfacheren Administration beider Datasets scheint der Betrieb als globale Ressource unter Produktionsbedingungen ratsam zu sein. Man kann sich dann nämlich auf bestehende Verfahren zum Benutzer-Management stützen und muss nicht eigene Verfahren zum Administrieren von Benutzern und deren Zugriffsmöglichkeiten entwickeln [mGRSp].
Mit Rücksicht auf den Einsatz als Schulungsobjekt wurde hier darauf verzichtet und lediglich eine Kopie der existierenden Dateien vorgenommen (siehe: 5.1.17 auf Seite 69).

Listing 5.27: JOB017 Teil 2 Synchronisieren BROADCAST<->UADS

```
//SYNC2    EXEC PGM=IKJEFT01
//*******************************************************************
//*          SYNC BRODCAST <-> UADS                                 *
//*******************************************************************
//SYSTSPRT  DD SYSOUT=*
//SYSLBC    DD DISP=SHR,UNIT=3390,VOL=SER=PEINS0,
//            DSN=SYS1.EINS.BRODCAST
//SYSUADS   DD DISP=SHR,UNIT=3390,VOL=SER=CEINS0,DSN=SYS1.EINS.UADS
//SYSTSIN   DD *
```

```
PROFILE NOPREFIX
ACCOUNT
SYNC UADS
END
//*-------------------------------------------------------------------
//DEFNVS3 EXEC PGM=IDCAMS
//*******************************************************************
//*          CATALOG UADS                                           *
//*******************************************************************
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  DEFINE NONVSAM(NAME(SYS1.EINS.UADS) DEVT(3390) VOLUMES(CEINS0)) -
         CATALOG(M.CEINS0)
```

5.1.18 Step 18 Unix-System-Services-Dateien einrichten

Natürlich muss auf einem Klon auch ein z/OS-Unix-Datei-System vorhanden sein. Die Unix-Komponente von z/OS heißt **Unix-System-Services (USS)**. Wie in jedem anderen Unix-Derivat muss ein hierarchisches Datei-System (Hierarchical File-System = HFS) vorhanden sein. Seit OS/390 gibt es zu diesem Zweck einen speziellen Dateityp namens 'HFS', um solche Unix-Dateisysteme aufzunehmen.
Ein HFS-Dataset muss auf einem SMS-verwalteten Volume angesiedelt sein und kann naturgemäß die maximale Größe des realen Volumes nicht überschreiten. Das sind im Falle einer 3390 Modell 3 Platte ca 2,8 GigaByte. Der Inhalt eines HFS-Datasets ist ein HFS-Unix-Dateisystem und kann nur durch die Unix-Betriebssystem-Komponente von „OS/390 Unix-System-Services"bzw. „Unix-for-zSeries" verändert werden.
Um USS- bzw. HSF-Datasets z/OS-seitig verwalten zu können, ist RACF oder ein äquivalentes Sicherheitssystem eines IBM-Mitbewerbers (z.B. CA Top Secret) nötig.
Die Inhalte mehrerer HFS-Datasets können hierarchisch miteinander verbunden werden. So ist es möglich, unter dem Root-Verzeichnis weitere Verzeichnisse in Form von HFS-Datasets zu „mounten" [zABC9].

Wie in jeder Installation gibt es sowohl system-spezifische HFS-Dateisysteme, auf die nur das besitzende System vollen Zugriff hat, als auch andere HFS-Dateisysteme, auf die ausschließlich „READ-ONLY" zugegriffen werden kann.
Zu den „READ-ONLY-" Datasets gehören die Datei 'RHFS.Z18.VERSION.HFS' und 'JVA140.HFS-Dataset'[3]. In der ersten Datei befinden sich sensible betriebssystem-relevante

[3] JAVA HFS Dataset

Informationen. Mit einem neuen Release des Betriebssystems werden auch neue Versions- und Java-HFS-Datasets augeliefert. Die alten HFS-Datasets werden einfach durch die neuen, moderneren Datasets ersetzt. Stellt sich heraus, das Programme mit der neuen Version nicht fehlerfrei laufen, so ist auch ein „Down-Cast“ jederzeit möglich.

Für jedes System werden auf der Platte HEINS0 folgende Datasets angelegt:

- `HFS.<System-Name>.ETC`
- `HFS.<System-Name>.VAR`
- `HFS.<System-Name>.TMP`
- `HFS.<System-Name>.U`

Dies geschieht durch das Kopieren bereits vorhandener HFS-Dateien von einem anderen z/OS-System mit dem DFDSS-Utility ADRDSSU (DFDSS = Data Facility Data Set Services).

Listing 5.28: JOB018 Aufbau HFS-Datasets

```
//DSCOPY2 EXEC PGM=ADRDSSU
//*********************************************************************
//*          DATASET COPY                                             *
//*********************************************************************
//SYSPRINT  DD SYSOUT=*
//Z8USS1    DD UNIT=3390,DISP=SHR,VOL=SER=Z8USS1
//HEINS0    DD UNIT=3390,DISP=SHR,VOL=SER=HEINS0
//DUMMY     DD DUMMY
//SYSIN     DD *
  COPY DATASET ( -
       INCLUDE   ( HFS.ADCD.DEV -
                   HFS.ADCD.ETC -
                   HFS.ADCD.TMP -
                   HFS.ADCD.VAR -
                   HFS.U.DB8G   ))  -
       RENAMEU   ( EINSM    ) INDDNAME ( Z8USS1) OUTDDNAME( HEINS0) -
       SHARE CATALOG STORCLAS (NONSMS) TGTALLOC (SOURCE) -
       TOLERATE (ENQFAILURE) WAIT(2,2)
//RENAME  EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
 ALTER  EINSM.ADCD.ETC   NEWNAME(HFS.EINS.ETC) CAT(M.CEINS0)
 ALTER  EINSM.ADCD.VAR   NEWNAME(HFS.EINS.VAR) CAT(M.CEINS0)
```

```
ALTER   EINSM.ADCD.DEV     NEWNAME(HFS.EINS.DEV)  CAT(M.CEINS0)
ALTER   EINSM.ADCD.TMP     NEWNAME(HFS.EINS.TMP)  CAT(M.CEINS0)
ALTER   EINSM.U.DB8G       NEWNAME(HFS.EINS.U  )  CAT(M.CEINS0)
```

5.1.19 Step 19 Erzeugen von SYS1.PARMLIB(LOAD00)

Mit dem IBM-Utility IEHUPDTE wird das - für das Starten des Systems unbedingt notwendige Member `'LOAD00'` - nach `'SYS1.PARMLIB'` geschrieben. Es enthält Verweise auf IODF-, NUCLST-, NUCLEUS-, SYSCAT-, SYSPARM- und IEASYM-Definitionen bzw. Informationen darüber.

Listing 5.29: JOB019 Einfügen SYS1.PARMLIB(LOAD00)

```
//ULISTORT JOB (ACCOUNT),'ULISTOR',MSGCLASS=X,MSGLEVEL=(1,1),
//             NOTIFY=&SYSUID,CLASS=A,REGION=6M
//*******************************************************************
//*    DOC: UPDATE SYS1.PARMLIB                                     ***
//*******************************************************************
//UPDTE1   EXEC PGM=IEBUPDTE,PARM=NEW
//*******************************************************************
//*          IEBUPDTE PARMLIB MEMBERS                               *
//*******************************************************************
//SYSPRINT  DD SYSOUT=*
//SYSUT2    DD DISP=SHR,UNIT=3390,VOL=SER=REINS0,DSN=SYS1.PARMLIB
//SYSIN     DD DATA,DLM='$$'
./ ADD NAME=LOAD00
IODF       00  IODF
NUCLST     00  N
NUCLEUS    1
SYSCAT     CEINS0113CM.CEINS0
SYSPARM    01
IEASYM     (00)
$$
//*------------------------------------------------------------------
//COMPRS2 EXEC PGM=IEBCOPY
//*******************************************************************
//*          COMPRESS SYS1.PARMLIB IN PLACE                         *
//*******************************************************************
//SYSPRINT  DD SYSOUT=*
//IN        DD UNIT=3390,VOL=SER=REINS0,DISP=SHR,DSN=SYS1.PARMLIB
//SYSIN     DD *
```

```
  COPY INDD=IN,OUTDD=IN
```

5.1.20 Step 20 Anlegen XCF-Couple-Datasets

In diesem Step werden die Couple-Datasets allokiert. Dazu gehören die XCF-Datasets (XCF = Cross Coupling Facility), die LOGR-Datsets und der WLM (Work-Load-Manager)-Datasets. Aus Performance-Gründen wird jedes dieser drei Datasets in doppelter Ausführung angelegt.

Prophylaktisch werden die Einträge für eventuell bereits vorhandene Datasets im Master-Katalog gelöscht, damit sie anschließend erzeugt, entsprechend formatiert und wieder im Master-Katalog eingetragen werden können. Falls beim vorsorglichen Löschen keine Dateien vorhanden sind, gibt IDCAMS in dieser Situation den Bedingungsschlüssel 12 zurück, obwohl eigentlich kein Fehler vorliegt. Daher wird unmittelbar nach dem Löschen der maximale Bedingungsschlüssel 'MAXCC'auf 0 gesetzt.

Listing 5.30: JOB020 Step 1 Löschen Couple-Dataset

```
//CATLG1   EXEC PGM=IDCAMS
//*******************************************************************
//*        UN-CATALOG COUPLE DATASETS                               *
//*******************************************************************
//SYSPRINT  DD SYSOUT=*
//DEINS0    DD DISP=OLD,UNIT=3390,VOL=SER=DEINS0
//SYSIN     DD *
  DEL SYS1.STORPLEX.CDS01.XCF  CAT(M.CEINS0) FILE(DEINS0)
  DEL SYS1.STORPLEX.CDS02.XCF  CAT(M.CEINS0) FILE(DEINS0)
  DEL SYS1.STORPLEX.CDS01.LOGR CAT(M.CEINS0) FILE(DEINS0)
  DEL SYS1.STORPLEX.CDS02.LOGR CAT(M.CEINS0) FILE(DEINS0)
  DEL SYS1.STORPLEX.CDS01.WLM  CAT(M.CEINS0) FILE(DEINS0)
  DEL SYS1.STORPLEX.CDS02.WLM  CAT(M.CEINS0) FILE(DEINS0)
  SET MAXCC=0
```

Nachdem sichergestellt ist, dass der Master-Katalog „sauber" ist, können die neuen Couple-Datasets angelegt werden.

Listing 5.31: JOB020 Step 2 Formatieren XCF-Couple-Datsets

```
//IXC2     EXEC PGM=IXCL1DSU
//*******************************************************************
//*        FORMAT SYSPLEX COUPLE DATASET                            *
//*******************************************************************
//SYSPRINT  DD SYSOUT=*
```

```
//SYSIN      DD *
  DEFINEDS DSN(SYS1.STORPLEX.CDS01.XCF) VOLSER(DEINS0)                    )
           MAXMEMBER(50) MAXGROUP(30) SYSPLEX(STORPLEX) STORCLAS(NONSMS)
  DEFINEDS DSN(SYS1.STORPLEX.CDS02.XCF) VOLSER(DEINS0)                    )
           MAXMEMBER(50) MAXGROUP(30) SYSPLEX(STORPLEX) STORCLAS(NONSMS)
  DEFINEDS SYSPLEX(STORPLEX) DSN(SYS1.STORPLEX.CDS01.LOGR) VOLSER(DEINS0)
           STORCLAS(NONSMS)
       DATA TYPE(LOGR)
           ITEM NAME(LSR) NUMBER(5)
           ITEM NAME(LSTRR) NUMBER(5)
           ITEM NAME(DSEXTENT) NUMBER(10)
  DEFINEDS SYSPLEX(STORPLEX) DSN(SYS1.STORPLEX.CDS02.LOGR) VOLSER(DEINS0)
           STORCLAS(NONSMS)
       DATA TYPE(LOGR)
           ITEM NAME(LSR) NUMBER(5)
           ITEM NAME(LSTRR) NUMBER(5)
           ITEM NAME(DSEXTENT) NUMBER(10)
  DEFINEDS SYSPLEX(STORPLEX) DSN(SYS1.STORPLEX.CDS01.WLM) VOLSER(DEINS0)
           MAXSYSTEM(2) STORCLAS(NONSMS)
       DATA TYPE(WLM)
           ITEM NAME(POLICY) NUMBER(3)
           ITEM NAME(WORKLOAD) NUMBER(35)
           ITEM NAME(SRVCLASS) NUMBER(30)
           ITEM NAME(APPLENV) NUMBER(100)
           ITEM NAME(SCHENV) NUMBER(100)
  DEFINEDS SYSPLEX(STORPLEX) DSN(SYS1.STORPLEX.CDS02.WLM) VOLSER(DEINS0)
           MAXSYSTEM(2) STORCLAS(NONSMS)
       DATA TYPE(WLM)
           ITEM NAME(POLICY) NUMBER(3)
           ITEM NAME(WORKLOAD) NUMBER(35)
           ITEM NAME(SRVCLASS) NUMBER(30)
           ITEM NAME(APPLENV) NUMBER(100)
           ITEM NAME(SCHENV) NUMBER(100)
```

Im folgenden werden **XCF**, **LOGR** und **WLM** kurz erläutert.

XCF - Cross-System Coupling Facility

Die XCF-Datasets ermöglichen bis zu 32 Systemen innerhalb eines Parallel-Sysplex, untereinander zu kommunizieren. XCF stellt dabei die Services zur Verfügung, mit denen parallelisierte Anwendungen Daten mit anderen derartigen Anwendungen auf einem anderen System austauschen können. Literatur dazu findet man in Kapitel 1.16 ‘Cross-system coupling facility (XCF)‘ in IBM Redbook ABCs of z/OS System Programming Volume 5 [zABC5].

Dabei werden folgende Services durch XCF bereitgestellt:

- **Group Services:**
  Die XCF Group Services ermöglichen es Programmen, eine Gruppe zu bilden und automatisch (über Exits) zu erfahren, wenn Gruppenmitglieder dazu kommen oder daraus entfernt werden. Dabei übernimmt XCF auch die Abmeldung, wenn ein beteiligtes Programm oder das System, auf dem es läuft, ohne Vorwarnung abstürzt.

- **Signaling-Services:**
  Die Signaling-Services von XCF dienen der Kommunikation von einzelnen Gruppenmitgliedern untereinander. Durch die Signaling-Services wird der Austausch der Nachrichten zwischen den Gruppenmitgliedern kontrolliert.
  Der Sender einer Nachricht fragt beim XCF-Signaling-Service den entsprechenden Service zwischen Sender und Empfänger an. XCF stellt nun einen Speicherbereich für die Nachrichten zwischen den Gruppenmitgliedern zur Verfügung. Durch die Signaling-Services werden die Nachrichten sysplex-weit verteilt.[zABC5]

- **Status-Monitoring-Services:**
  Die Status-Monitoring-Services sind sog. „authorized services“ also sehr mächtige Dienste, die im Administrator-Modus ausgeführt werden können. Diese Dienste zeichnen die Statusänderungen von Gruppenmitgliedern auf. Eine Statusänderung tritt immer dann auf, wenn eine „Exit-Routine“ ausgeführt wird auf. Gruppenmitglieder können auf Anfrage bei XCF Ihre Statusänderungen aufzeichnen lassen [zABC5].

Detaillierte Informationen zum diesem Bereich gibt es in Kapitel 1.19 ‘XCF services‘in IBM Redbook ABCs of z/OS System Programming Volume 5 [zABC5].

LOGR - LOGR-Couple Datasets
Jedes System in einem Parallel-Sysplex hat seinen eigenen Logging-Service. Der Logging-Service besteht aus dem LOGREC- und dem LOGR (Syslog). Beide Services laufen unabhängig von einander und nutzen ihre eigenen Datasets.
Das LOGR-Couple-Dataset (CDS) hält Informationen zu der System-Logger-Policy und Informationen über alle definierten LOG-Streams vor. Das LOGR-CDS muss für alle Syteme Sysplexes zugänglich sein. Durch ein globales LOGR-CDS ist es möglich, die Log-Streams mit einem globalen Zeitstempel auszustatten oder alle Log-Einträge innerhalb des Sysplexes sequenziell zu betrachten [sABC5].

WLM - Work-Load Manager

Der Workload Manager ist in jedem System vorhanden. Um die in einem System anfallende Last gleichmäßig in einem Parallel-Sysplex verteilen zu können, ist es nötig, dass die einzelnen WLMs miteinander kommunizieren können. Der zentrale Punkt zum Austausch von Informationen ist natürlich die Coupling-Facility, die über die WLM-Couple-Datasets die entsprechenden Ressourcen zur Verfügung stellt, um alle WLMs des Sysplexes miteinander zu verbinden.

Der WLM arbeitet ensprechend den Vorgaben, die seitens der Systemadministration eingestellt werden. Dabei können die Ziele, die der WLM erreichen soll, Nebenbedingungen verschiedenster Art haben. So kann beispielsweise der Schwerpunkt des Workloads nachts auf die Verarbeitung von Long-Running-Jobs liegen, während der Schwerpunkt tagsüber auf den Online-Betrieb (CICS, DB2, IMS, TSO etc.) gelegt wird.

Um an die Statusinformationen (momentane Auslastung, Speicherbelegung, angeschlossene I/O Devices, etc.) des Systems zu gelangen, ist dem WLM eine weitere Komponente namens SRM (System Resource Manager) unterstellt, die diese Informationen fortlaufend sammelt. Anhand dieser Informationen werden die Strategien festgelegt, die die laufenden Applikationen auf die vorhandenen Ressourcen aufteilen. Dazu werden die Arbeitsanforderungen in Service-Klassen aufgeteilt. Durch diese Klassen werden den Arbeitsanforderungen unter anderem auch Prioritäten zugeordnet. Bei der Bearbeitungsreihenfolge werden die Aufträge, mit der höchsten Priorität als erstes erledigt.

Die Kommunikation zwischen den einzelnen Systemen erfolgt über die WLM-Couple-Datasets auf der CF, im speziellen über die Cache- und die Locking-Strukturen. [zABC11]

Zum Schluss sind die erzeugten Dateien noch zu katalogisieren:

Listing 5.32: JOB020 Step 3 Katalogisieren XCF-Couple-Datsets

```
//CATLG3   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//DEINS0    DD DISP=OLD,UNIT=3390,VOL=SER=DEINS0
//SYSIN     DD *
  DEF NVSAM(NAME(SYS1.STORPLEX.CDS01.XCF                    ) +
      VOL(DEINS0) DEVT(3390)) CAT(M.CEINS0)
  DEF NVSAM(NAME(SYS1.STORPLEX.CDS02.XCF                    ) +
      VOL(DEINS0) DEVT(3390)) CAT(M.CEINS0)
  DEF NVSAM(NAME(SYS1.STORPLEX.CDS01.LOGR                   ) +
      VOL(DEINS0) DEVT(3390)) CAT(M.CEINS0)
  DEF NVSAM(NAME(SYS1.STORPLEX.CDS02.LOGR                   ) +
      VOL(DEINS0) DEVT(3390)) CAT(M.CEINS0)
  DEF NVSAM(NAME(SYS1.STORPLEX.CDS01.WLM                    ) +
```

```
        VOL(DEINS0) DEVT(3390)) CAT(M.CEINS0)
    DEF NVSAM(NAME(SYS1.STORPLEX.CDS02.WLM                    ) +
        VOL(DEINS0) DEVT(3390)) CAT(M.CEINS0)
```

5.2 Anpassungen des Masterklons an einen Sysplex-Betrieb

5.2.1 MAS - Multi-Access-Spool

Durch den gemeinsam genutzten Spoolbreich ist es möglich, Jobs, die von einem System abgeschickt (submitted) wurden, auf einem anderen System innerhalb des Sysplexes ausführen zu lassen.
Notwendigerweise erfordert ein MAS, dass alle JES2-Checkpoint-Datasets sich auf dem gleichen Speicher befinden. Dies kann ein normales Volume sein oder aber ein Speicherbereich, der sich auf der Coupling-Facility befindet.
Ein Multi-Access-Spool besteht aus zwei oder mehr JES2-Instanzen, die sich die gleichen Checkpoint-Datasets teilen. Es gibt keine direkte Verbindung zwischen den einzelnen JES2- Instanzen. Die Kommunikation zwischen den einzelnen Instanzen wird durch das DASD implementiert, auf welchem sich die Checkpointdatasets befinden. Jedes JES2 in einem Parallel-Sysplex funktioniert unabhängig von den anderen Instanzen. In einem JES2-Verbund teilen sich die Systeme über die Checkpoint-Datasets eine gemeinsame Job-Queue und eine gemeinsame Output-Queue. Durch das Teilen dieser zwei Puffer wird es ermöglicht, die Arbeitslast der einzelnen Systeme über den gesamten Parallel-Sysplex nach Bedarf zu verteilen. So können Jobs auf einem System angestoßen werden, und den Output auf einem anderen System ablegen, welches die nötigen Vorausetzungen hat (z.B. offener Initiator für die geforderte Job-Klasse). Der Benutzer kann auch festlegen, auf welchem System der Job ausgeführt werden soll und auf welchem System er den Output erwartet. Sollte es zu einem Ausfall eines Systems kommen, so können die verbleibenden Systeme den zusätzlichen Workload unter sich aufteilen. Es gehen nur wenige Daten verloren, nämlich nur die, die gerade auf dem betreffenden System bearbeitet wurden.

5.2.2 BPXPRMS1: Anpassungen für das Unix-Segment

Im SYS1.PARMLIB-Member BPXPRMxx wird das Unix-Segment von z/OS mit entsprechenden Parametern beschrieben. Hier wird auch festgelegt, wie das Dateisystem aufgebaut wird:
Mit der Anweisung `‘ROOT FILESYSTEM(’HFS.Z18.VERSION.HFS’) TYPE(HFS) MODE(READ)‘`

wird bestimmt, dass das z/OS-HFS-Dataset ‘HFS.Z18.VERSION.HFS‘ das Root-Verzeichnis enthalten soll. Die HFS-Datei wurde ebenso wie die folgenden Dateien in Step 18 erzeugt und diese stellen nun die auch aus anderen Unix-Systemen bekannten Verzeichnisse dar. Unter dem Root-Verzeichnis wird ausßerdem das Java-File-System angeschlossen. Das entsprechende Dataset heißt `'JVA140.HFS'`.
Hier ein Auszug aus BPXPRMS1 (`&SYSNAME` wurde durch unseren Systemnamen 'EINS' ersetzt):

Listing 5.33: Extrakt aus SYS1.PARMLIB(BPXPRMS1)

```
ROOT  FILESYSTEM('HFS.Z18.VERSION.HFS') TYPE(HFS) MODE(READ)
MOUNT FILESYSTEM('HFS.EINS.DEV') TYPE(HFS) MODE(RDWR) MOUNTPOINT('/dev')
MOUNT FILESYSTEM('HFS.EINS.VAR') TYPE(HFS) MODE(RDWR) MOUNTPOINT('/var')
MOUNT FILESYSTEM('HFS.EINS.ETC') TYPE(HFS) MODE(RDWR) MOUNTPOINT('/etc')
MOUNT FILESYSTEM('HFS.EINS.TMP') TYPE(HFS) MODE(RDWR) MOUNTPOINT('/tmp')
MOUNT FILESYSTEM('HFS.EINS.U') TYPE(HFS) MODE(RDWR) MOUNTPOINT('/u')
MOUNT FILESYSTEM('JVA140.HFS') TYPE(HFS) MODE(READ) MOUNTPOINT('/usr/lpp/java')
}
```

Danach sieht das z/OS-Unix-Datei-System so aus:

Listing 5.34: Abbildung des ‘Hierarchical File-System‘

```
/
/dev
/etc
/tmp
/u
/usr/lpp/java
/var
```

5.2.3 COUPLESx: Couple-Datasets im System bekanntmachen

Es gibt in dieser Installation zwei Varianten, den Sysplex zu betreiben. Die Erste wird durch den Suffix „S0“ am Ende der relevanten Datasets bzw. Datei-Member, die Zweite durch den Suffix „S1“ gekennzeichnet. Diese Variante beinhaltet auch die Allokierung und Nutzung des CFRM-Couple-Datasets (CFRM-Coupling Facility Resource Manager).
In den Members COUPLES0 und/oder COUPLES1 werden die entsprechenden Couple-Datasets eingetragen. Für die Dienste XCF, LOGR, WLM und CFRM wird je ein Paar Dateien erstellt. Paar bedeutet hier: Es gibt jeweils ein primäres und ein sekundäres/alternatives Dataset.
Es wird empfohlen, um einen „Single-Point-of-Failure“ zu vermeiden, die Couple-Datasets

(CDS) auf getrennten Volumes und über von einander getrennte Kanäle anzulegen bzw anzusteuern. Administrative Änderungen auf dem primären CDS werden auch auf dem sekundären CDS umgehend durchgeführt.
Im Falle eines Ausfalls des primären CDS wird automatisch das sekundäre CDS zum primären CDS umgeschaltet. Bei einem manuellen Umschalten wird durch den entsprechenden Befehl das ehemals primäre CDS nicht zum sekundären CDS, sondern wird schlicht deallokiert.
Dadurch werden folgenden Operationen ermöglicht:

- Änderung der Größe des primären CDS.
  → Nach dem das primäre CDS wieder allokiert worden ist, wird automatisch auch die Größe des sekundären CDS geändert.
- Verlegen des deallokierten CDS auf ein anderes Laufwerk
- Änderung der CDS Definitionen
- Formatierung des CDS

Listing 5.35: Extrakt aus SYS1.PARMLIB(COUPLES1)

```
/* ------------------------------------------------------------------ */
/* Member COUPLES1: &SYSPLEX wird ersetzt z.b. durch STORPLEX          */
/* ------------------------------------------------------------------ */
COUPLE SYSPLEX(&SYSPLEX)
       PCOUPLE(SYS1.&SYSPLEX..CDS01.XCF)    /* XCF-Prim.Couple-DS   */
       ACOUPLE(SYS1.&SYSPLEX..CDS02.XCF)    /* XCF-Alt.-Couple-DS   */
   DATA TYPE(LOGR)
       PCOUPLE(SYS1.&SYSPLEX..CDS01.LOGR)   /* LOGR-Prim.Couple-DS  */
       ACOUPLE(SYS1.&SYSPLEX..CDS02.LOGR)   /* LOGR-Alt.-Couple-DS  */
   DATA TYPE(WLM)
       PCOUPLE(SYS1.&SYSPLEX..CDS01.WLM)    /* WLM-Prim.Couple-DS   */
       ACOUPLE(SYS1.&SYSPLEX..CDS02.WLM)    /* WLM-Alt.-Couple-DS   */
   DATA TYPE(CFRM)
       PCOUPLE(SYS1.&SYSPLEX..CDS01.CFRM)   /* CFRM-Prim.Couple-DS  */
       ACOUPLE(SYS1.&SYSPLEX..CDS02.CFRM)   /* CFRM-Alt.-Couple-DS  */
/*                                                                    */
/* XCF-Signaling-Pathes Inbound/Outbound                             */
/* ------------------------------------                               */
PATHIN  DEVICE(&PI01.,&PI02.)
PATHOUT DEVICE(&PO01.,&PO02.)
PATHIN  STRNAME(IXCLST01,IXCLST02)
PATHOUT STRNAME(IXCLST01,IXCLST02)
```

5.2.4 IEASYSxx: Setzen von Variablen für die System-Initialisierung

Das `IEASYSxx`-Member der `SYS1.PARMLIB` beinhaltet einige Variablen, die zur Initialisierung des Systems benötigt werden. Wie in dem vorangegangen Abschnitt erklärt gibt es auch beim `IEASYSxx`-Member zwei Varianten:

S0 : Das Member `IEASYSS0` startet u.a. die Couple-Konfiguration „S0“ und USS mit `BPXPRMS1` (vergl. OMVS=S1)

S1 : Das Member `IEASYSS1` startet u.a. die Couple-Konfiguration „S1“ und USS mit `BPXPRMS1` wie in `IEASYSS0`

Nachfolgend ein Auszug aus IEASYSS1:

Listing 5.36: Extrakt aus SYS1.PARMLIB(IEASYSS1)

```
/* ------------------------------------------------------------------ */
/* Member IEASYSS1: System Parameter List                             */
/* ------------------------------------------------------------------ */
/* Zentrales Member zur Steuerung des System-Starts                   */
/* verweist vielfach auf andere Start-Member mit spezif. Parametern   */
/* z.B. COUPLE=S1 zeigt auf Memmer COUPLES1                           */
/* !!! Es wird nur ein Ausschnitt gezeigt !!!                         */
/* ------------------------------------------------------------------ */
...
CON=(00,NOJES3),                  SELECT CONSOL00
COUPLE=S1,                        USE COUPLES1
CSA=(3000,200000),                CSA RANGE
LNKAUTH=LNKLST,                   AUTHORIZE LNKLST00, APFTAB IS ALTERNATE
LOGREC=SYS1.&SYSNAME..LOGREC,     ERROR RECORDING
LPA=S1,                           SELECT LPALST00
MAXUSER=250,                      SYS TASKS PLUS INITS PLUS TSOUSERS
MSTRJCL=S1,                       SELECT MSTJCLEX, MASTER JCL
OMVS=S1,                          SELECT BPXPRMCS
PAGE=(SYS1.&SYSNAME..PAGE.PLPA,
      SYS1.&SYSNAME..PAGE.COMMON,
      SYS1.&SYSNAME..PAGE.LOCAL1,L),
SMF=00,                           SELECT SMFPRM00, SMF PARAMETERS
SSN=(02,00),                      SELECT IEFSSN00, SUBSYSTEM NAMES
SVC=00,                           SELECT IEASVC00, USER SVCS
VIODSN=SYS1.&SYSNAME..STGINDEX,   VIO DS
VRREGN=64                         DEFAULT REAL-STORAGE REGION SIZE
```

5.2.5 JES2PARM: JES für den MAS-Betrieb einrichten

Der später entstehende Parallel-Sysplex wird mit einem Multi-Access-Spool betrieben. Wird ein Klon im Single-Modus gestartet, so bleibt der MAS als solches erhalten, er wird aber nur von einem System benutzt (6.2.2 auf Seite 94).

Listing 5.37: Extrakt aus SYS1.PARMLIB(JES2PARM) with MAS-Setup

```
/* ------------------------------------------------------------------ */
/* Member JES2PARM                                                    */
/* ------------------------------------------------------------------ */
/* CKPTDEF-Anweisung zeigt auf die Checkpoint-Datasets:               */
/*          SYS1.&SYSPLEX..HASPCK01 bzw.                              */
/*          SYS1.&SYSPLEX..HASPCK02 (siehe JOB009)                    */
/* MASDEF-Anweisung zur Einrichtung des MULTI ACCESS SPOOL            */
/* SPOOLDEF schafft die Verbindung zum Spool-Dataset:                 */
/*          SYS1.&SYSPLEX..HASPACE  (siehe JOB009)                    */
/* ------------------------------------------------------------------ */
CKPTDEF  CKPT1=(DSNAME=SYS1.&SYSPLEX..HASPCK01,VOLSER=PEINS0,INUSE=YES),
          NEWCKPT1=(DSNAME=SYS1.&SYSPLEX..HASPCKN1,VOLSER=PEINS0),
         CKPT2=(DSNAME=SYS1.&SYSPLEX..HASPCK02,VOLSER=PEINS0,INUSE=YES),
          LOGSIZE=2,VERSIONS=(NUMBER=0,WARN=80),
          DUPLEX=ON,MODE=DUPLEX,VOLATILE=(ONECKPT=WTOR,ALLCKPT=WTOR)
/*********************************************************************/
INITDEF  PARTNUM=12
INIT(1)  NAME=1,CLASS=A,START
...
INIT(12) NAME=12,CLASS=ABCD,START=NO
...
/*********************************************************************/
MASDEF   SHARED=NOCHECK,                  /* MULTI ACCESS SPOOL NOCHECK */
          DORMANCY=(100,500),HOLD=100,LOCKOUT=1200,
          SID(1)=SY01,
          SID(2)=SY02,
          SYNCTOL=120
/*                                                                    */
SPOOLDEF BUFSIZE=3856,SSNAME=SYS1.&SYSPLEX..HASPACE,VOLUME=PEINS,
          FENCE=NO,SPOOLNUM=32,TGBPERVL=5,TGSIZE=33,
          TGSPACE=(MAX=26288,TRKCELL=3
/*... viele weitere Anweisungen
```

Dieser Auszug aus dem Member JES2PARM zeigt die Definition des Multi-Access-Spool für zwei Systeme.

5.2.6 LOADxx: Setzen der System-Start-Parameter

Das Member LOADxx wird als eines der ersten beim IPL angezogen. Durch die Auswahl des Suffixes „S0“ oder „S1“ kann das System bei IPL in verschiedenen Modi wie oben beschrieben gestartet werden. LOADS0 beinhaltet z.B. folgende Informationen:

Listing 5.38: Das Member SYS1.PARMLIB(LOADS0)

```
* -------- *
* LOADS0   *
* ---------+-------------------------------------------------------------
* Der Aufbau von LOADxx ist spaltengerecht,
* Ab Sp. 1 Parameter, ab Sp. 10 Daten, Kommentar = '*' in Sp. 1
* ------------------------------------------------------------------------
* IODF zeigt auf: IODF.IODF02
* NUCLST  zeigt auf SYS1.PARMLIB(NUCLST00) mit NOWAIT
* NUCLEUS zeigt auf SYS1.NUCLEUS(IEANUC1)
* SYSCAT  zeigt auf CEINS0 = MCAT-Volume, 1 = SYS1-Konversion aktiv,
*                    1 = Aliaslevel, 3C = CAS Serv.-Level,
*                   M.CEINS0 = Name des Master-Katalogs
* SYSPARM zeigt auf SYS1.PARMLIB(IEASYSS0)
* IEASYM  zeigt auf SYS1.PARMLIB(IEASYSS0) und SYS1.PARMLIB(IEASYSSY)
* SYSPLEX setzt den Namen des Sysplex auf STORPLEX
* ------------------------------------------------------------------------
IODF      02 IODF
NUCLST    00 N
NUCLEUS   1
SYSCAT    CEINS0113CM.CEINS0
SYSPARM   S0
IEASYM    (S0,SY)
SYSPLEX   STORPLEX
```

Listing 5.39: Das Member SYS1.PARMLIB(LOADS1)

```
* -------- *
* LOADS1   *
* ---------+-------------------------------------------------------------
* Der Aufbau von LOADxx ist spaltengerecht,
* Ab Sp. 1 Parameter, ab Sp. 10 Daten, Kommentar = '*' in Sp. 1
* ------------------------------------------------------------------------
* IODF zeigt auf: IODF.IODF02
* NUCLST  zeigt auf SYS1.PARMLIB(NUCLST00) mit NOWAIT
* NUCLEUS zeigt auf SYS1.NUCLEUS(IEANUC1)
* SYSCAT  zeigt auf CEINS0 = MCAT-Volume, 1 = SYS1-Konversion aktiv,
*                    1 = Aliaslevel, 3C = CAS Serv.-Level,
```

```
*                    M.CEINS0 = Name des Master-Katalogs
* SYSPARM zeigt auf SYS1.PARMLIB(IEASYSS1)
* IEASYM  zeigt auf SYS1.PARMLIB(IEASYSS1) und SYS1.PARMLIB(IEASYSSY)
* SYSPLEX setzt den Namen des Sysplex auf STORPLEX
* ------------------------------------------------------------------
IODF      02 IODF
NUCLST    00 N
NUCLEUS   1
SYSCAT    CEINS0113CM.CEINS0
SYSPARM   S1
IEASYM    (S1,SY)
SYSPLEX   STORPLEX
```

6 Das Klonen vom Master-System

6.1 Vorgehensweise

6.1.1 Step 01 Paging- und Spooling-Dateien einrichten

Um den zweiten Klon zu erstellen, müssen eine weitere Paging-Platte und eine weitere Spooling-Platten bereitgestellt werden. Dies wird mit dem Programm „ICKDSF “ und dem Befehl:
`INIT NOCHK NOVFY NOVAL PURGE VTOC(0,3,12) INDEX(0,1,2) UNIT(306) VOLID(PZWEI0)`
realisiert. Hierbei ist darauf zu achten, dass das Volume zuvor mit
`VARY 306,OFFLINE`
für den normalen Betrieb gesperrt wird. Nach der Initialisierung wird das Volume mit
`VARY 306,ONLINE`
wieder aktiviert.

Listing 6.1: JOB101 Klon->Klon Initialisieren Platte

```
//*    ATTENTION CHECK DEVICE-NUMBER                                  ***
//*    FIRST: SET DEVICE    OFFLINE (VARY 306,OFFLINE)                ***
//*    SECON: RUN JOB101, ANSWER REQUESTS WITH 'U'                    ***
//*    THIRD: SET DEVICE   ONLINE (VARY 306,ONLINE)                   ***
//*******************************************************************
//*-----------------------------------------------------------------
//INITDSK EXEC PGM=ICKDSF
//*******************************************************************
//*         CREATE VTOC & INDEX                                      *
//*******************************************************************
//SYSPRINT  DD SYSOUT=*
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
 INIT NOCHK NOVY PRG VTOC(0,3,12) INDEX(0,1,2) UNIT(306) VOLID(PZWEI0)
```

6.1.2 Step 02 Anlegen Page-Datasets, STGINDEX und SMF-Dateien

Auf der Paging-Platte werden wie schon für das erste System drei Page-Spaces (PLPA, COMMON, LOCAL) installiert - siehe oben Punkt 5.1.5 auf Seite 53). Der folgende Job wird nur auszugsweise dargestellt:

Listing 6.2: JOB102 Step 1 Klon->Klon Aufbau Page-Spaces

```
//PAGEDS1 EXEC PGM=IDCAMS
//*******************************************************************
//*          PAGESPACE                                              *
//*******************************************************************
//SYSPRINT  DD SYSOUT=*
//PZWEI0    DD UNIT=3390,VOL=SER=PZWEI0,DISP=SHR
//SYSIN     DD *
  DEFINE PAGESPACE (NAME(EINSM.SYS1.ZWEI.PAGE.PLPA) FILE(PZWEI0) -
                CYLINDERS(1)    VOLUME(PZWEI0) STORCLAS(NONSMS) UNIQUE)
  DEFINE PAGESPACE (NAME(EINSM.SYS1.ZWEI.PAGE.COMMON) FILE(PZWEI0) -
                CYLINDERS(299) VOLUME(PZWEI0) STORCLAS(NONSMS) UNIQUE)
  DEFINE PAGESPACE (NAME(EINSM.SYS1.ZWEI.PAGE.LOCAL1) FILE(PZWEI0) -
                CYLINDERS(400) VOLUME(PZWEI0) STORCLAS(NONSMS) UNIQUE)
 ALTER EINSM.SYS1.ZWEI.PAGE.PLPA          NEWNAME(SYS1.ZWEI.PAGE.PLPA)
  ...
 DEFINE CLUSTER (NAME(SYS1.ZWEI.STGINDEX) FILE(PZWEI0) -
                KEYS(12,8) CYLINDERS(2) RECORDSIZE(2041,2041) REUSE -
                VOLUME(PZWEI0) STORCLAS(NONSMS) BUFFERSPACE(20480) ) -
  DATA          (CONTROLINTERVALSIZE(2048)) -
  INDEX         (CONTROLINTERVALSIZE(1024)) -
  CATALOG       (M.CEINS0)
  ...
  DEFINE CLUSTER (NAME(SYS1.ZWEI.MAN1) FILE(PZWEI0) -
                VOLUME(PZWEI0) STORCLAS(NONSMS) CYLINDERS(30)  -
                RECORDSIZE(4086,32767) CONTROLINTERVALSIZE(4096) -
                SHAREOPTIONS(2) NONINDEXED SPANNED REUSE SPEED) -
  CATALOG       (M.CEINS0)
  DEFINE CLUSTER (NAME(SYS1.ZWEI.MAN2) FILE(PZWEI0) -
  ...
```

6.1.3 Step 03 LOGREC- und BROADCAST-Datasets einrichten

Im nächsten Schritt werden mit dem Programm „IFCDIP00“ das LOGREC-Dataset eingerichtet, mit „IEFBR14“ das BROADCAST-Dataset allokiert und zum Schluss die beiden Datasets mit dem Programm „IDCAMS“ im Master-Katalog eingetragen.

Das entspricht der Vorgehensweise unter „Step 10“ in 5.1.10 auf Seite 62; daher wird JOB103 verkürzt dargestellt:

Listing 6.3: JOB103 Step 1 Klon->Klon Aufbau LOGREC-Dataset

```
//LOGREC1 EXEC PGM=IFCDIP00
//*********************************************************************
//*          INIT  SYS1.LOGREC                                        *
//*********************************************************************
//SERERDS   DD DSN=EINSM.SYS1.ZWEI.LOGREC,DISP=(NEW,CATLG),UNIT=3390,
//             VOLUME=(,RETAIN,SER=PZWEI0),STORCLAS=NONSMS,
//             SPACE=(CYL,5)
...
//BRODCS2 EXEC PGM=IEFBR14
//*********************************************************************
//*          ALLOC EINSM.SYS1.BRODCAST,  DAE                          *
//*********************************************************************
//SYSUT2    DD DISP=(,CATLG),UNIT=3390,VOL=SER=PZWEI0,
//             SPACE=(CYL,1),STORCLAS=NONSMS,
//             DCB=(LRECL=00129,BLKSIZE=00129,RECFM=F),
//             DSN=EINSM.SYS1.ZWEI.BRODCAST
...
//CATALOG EXEC PGM=IDCAMS
//*********************************************************************
//*          CATALOG  IN  M.CEINS0                                    *
//*********************************************************************
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
 ALTER EINSM.SYS1.ZWEI.BRODCAST -
             NEWNAME(SYS1.ZWEI.BRODCAST)  CAT(M.CEINS0)
 ALTER EINSM.SYS1.ZWEI.LOGREC -
       NEWNAME(SYS1.ZWEI.LOGREC)  CAT(M.CEINS0)
```

6.1.4 Step 04 VTAM-APPN-Datasets einrichten

Die VTAM- und APPN-Dateien sind für den neuen Klon zu generieren.

Informationen über VTAM und das Einrichten der Topologie-Datasets findet man unter „Step 11“ in 5.1.11 auf Seite 63.

Listing 6.4: JOB104 Klon->Klon Anlegen VTAM/APPN-Datasets

```
//*    DOC:  ALLOCATE + CATALOG VTAM APPN TOPOLOGY DATASETS          ***
//*********************************************************************
//ALLOCATE  EXEC PGM=IEFBR14
```

```
//DD1          DD DISP=(,KEEP),DSN=SYS1.ZWEI.DSDBCTRL,UNIT=3390,
//                VOL=SER=PZWEI0,SPACE=(CYL,(1,1)),STORCLAS=NONSMS,
//                DCB=(LRECL=20,BLKSIZE=20,RECFM=F)
//DD2          DD DISP=(,KEEP),DSN=SYS1.ZWEI.DSDB1,UNIT=3390,
//                VOL=SER=PZWEI0,SPACE=(CYL,(1,1)),STORCLAS=NONSMS,
//                DCB=(LRECL=1000,BLKSIZE=27000,RECFM=FB)
//DD3          DD DISP=(,KEEP),DSN=SYS1.ZWEI.DSDB2,UNIT=3390,
//                VOL=SER=PZWEI0,SPACE=(CYL,(1,1)),STORCLAS=NONSMS,
//                DCB=(LRECL=1000,BLKSIZE=27000,RECFM=FB)
//DD4          DD DISP=(,KEEP),DSN=SYS1.ZWEI.TRSDB,UNIT=3390,
//                VOL=SER=PZWEI0,SPACE=(CYL,(1,1)),STORCLAS=NONSMS,
//                DCB=(LRECL=1000,BLKSIZE=27000,RECFM=FB)
//*
//CATALOG    EXEC PGM=IDCAMS
//SYSPRINT   DD SYSOUT=*
//SYSIN      DD *
  DEF NVSAM (NAME(SYS1.ZWEI.DSDBCTRL) VOL(PZWEI0) DEVT(3390)) -
             CAT(M.CEINS0)
  DEF NVSAM (NAME(SYS1.ZWEI.DSDB1) VOL(PZWEI0) DEVT(3390)) -
             CAT(M.CEINS0)
  DEF NVSAM (NAME(SYS1.ZWEI.DSDB2) VOL(PZWEI0) DEVT(3390)) -
             CAT(M.CEINS0)
  DEF NVSAM (NAME(SYS1.ZWEI.TRSDB) VOL(PZWEI0) DEVT(3390)) -
             CAT(M.CEINS0)
```

6.1.5 Step 05 UA-Dataset einrichten und mit BROADCAST-Datei synchronisieren

Für jeden weiteren Klon werden ein weiteres UADS- und BROADCAST-Dataset benötigt. Die Vorgehensweise entspricht der Beschreibung unter Punkt 5.1.17 auf Seite 69.

Listing 6.5: JOB105 Klon->Klon Aufbau UADS/Broadcast-Datasets

```
//*    DOC: COPY,SYNC UADS                                          ***
//DEFNVS3 EXEC PGM=IDCAMS
//*                                                  DELETE UADS    ***
//CEINS0     DD UNIT=3390,VOL=SER=CEINS0,DISP=SHR
//SYSPRINT   DD SYSOUT=*
//SYSIN      DD *
  DELETE SYS1.ZWEI.UADS CATALOG(M.CEINS0) FILE(CEINS0)
  SET MAXCC=0
//*
//COPYUA1 EXEC PGM=IEBCOPY,REGION=6M
//*                                        COPY SYS1.UADS TO CEINS0  ***
```

```
//SYSPRINT  DD SYSOUT=*
//IN        DD DSN=SYS1.UADS,DISP=SHR
//OUT       DD DSN=SYS1.ZWEI.UADS,UNIT=3390,VOL=SER=CEINS0,
//             DISP=(,KEEP,DELETE),
//             SPACE=(CYL,(1,1,20)),
//             STORCLAS=NONSMS,
//             DCB=(LRECL=80,RECFM=FB,BLKSIZE=800)
//SYSIN     DD *
 COPY INDD=IN,OUTDD=OUT
//*
//SYNC2    EXEC PGM=IKJEFT01
//*                                          SYNC BRODCAST <-> UADS   ***
//SYSTSPRT  DD SYSOUT=*
//SYSLBC    DD DISP=SHR,UNIT=3390,VOL=SER=PEINS0,DSN=SYS1.ZWEI.BRODCAST
//SYSUADS   DD DISP=SHR,UNIT=3390,VOL=SER=CEINS0,DSN=SYS1.ZWEI.UADS
//SYSTSIN   DD *
PROFILE NOPREFIX
ACCOUNT
SYNC UADS
END
//DEFNVS3 EXEC PGM=IDCAMS
//*                                                CATALOG UADS     ***
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
  DEFINE NONVSAM(NAME(SYS1.ZWEI.UADS) DEVT(3390) VOLUMES(CEINS0)) -
         CATALOG(M.CEINS0)
```

6.1.6 Step 06 USS-HFS-Filesystem erzeugen

Hier wird das Unix-System-Services-Dateisystem (HFS-Filessystem) erzeugt (siehe auch 5.1.18 auf Seite 71 ff.).

Listing 6.6: JOB106 Klon->Klon Kopieren HFS-Datasets

```
//*        DATASET COPY                                                *
//SYSPRINT  DD SYSOUT=*
//Z8USS1    DD UNIT=3390,DISP=SHR,VOL=SER=Z8USS1
//HEINS0    DD UNIT=3390,DISP=SHR,VOL=SER=HEINS0
//DUMMY     DD DUMMY
//SYSIN     DD *
  COPY DATASET ( -
       INCLUDE   ( HFS.ADCD.DEV -
                   HFS.ADCD.ETC -
                   HFS.ADCD.TMP -
```

```
                    HFS.ADCD.VAR -
                    HFS.U.DB8G      )) -
        RENAMEU  ( EINSM    ) -
        INDDNAME ( Z8USS1) OUTDDNAME( HEINS0) -
        SHARE CATALOG STORCLAS (NONSMS) TGTALLOC (SOURCE) -
        TOLERATE (ENQFAILURE) WAIT(2,2)
//*
//*        CATALOG DATASET                                              *
//RENAME  EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
 ALTER  EINSM.ADCD.ETC   NEWNAME(HFS.ZWEI.ETC) CAT(M.CEINS0)
 ALTER  EINSM.ADCD.VAR   NEWNAME(HFS.ZWEI.VAR) CAT(M.CEINS0)
  ...
```

6.1.7 Step 07 System-Namen vergeben

Im letzten Schritt wird das neue System „getauft“. Dazu wird mit Hilfe der von IBM bereitgestellten Prozedur „ACBJBAOB“ ein Systemname festgelegt und dieser im Sysplex bekanntgegeben.

Listing 6.7: JOB107 Klon->Klon Umbenennen System-Name in SCDS

```
//BEGEL007 JOB (ACCOUNT),'BEGEL',MSGCLASS=X,MSGLEVEL=(1,1),
//           NOTIFY=&SYSUID,CLASS=A,REGION=6M
//*****************************************************************
//*    JOB  SUBMITTED FROM BEGEL.SYSPLEX.CNTL(JOB107)              ***
//*         RENAME SYSTEMNAME IN SCDS WITH NAIQUEST                ***
//*****************************************************************
//RENSYS  EXEC ACBJBAOB
//STEP2.SYSUDUMP DD  SYSOUT=*
//STEP2.SYSTSIN  DD *
BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(999999)
ISPSTART CMD(ACBQBAB1 ALTER SCDS('EINS1.SMS.SYS1PLEX.SCDS') +
             ADDSYS(SYS1Z02) ) +
BATSCRW(132) BATSCRD(27) BREDIMAX(3) BDISPMAX(999999)
```

6.2 Sytemanpassungen für jedes weitere System

Um das neue System in den Parallel-Sysplex zu integrieren, müssen in der SXS1.PARMLIB in zwei Membern die Einstellungen angepasst werden.

6.2.1 IEASYMS1: Beschreibung der Verbindungen zwischen den Systemen

Im Member IEASYMS1 von SYS1.PARMLIB werden die CTC Verbindungen zwischen den Systemen innerhalb des Sysplexes angegeben. Für jedes System werden jeweils zwei Ausgangs- und zwei Eingangs-Kanäle definiert. Bei einem Parallel-Sysplex ist die Verwaltung der Schnittstellen relalativ einfach. Mit steigender Zahl der Systeme wird jedoch die Komplexität immer größer.
Wie Abbildung 6.1 auf Seite 93 zeigt, hat jedes Sytem zu jedem anderen jeweils zwei Eingangskanäle und zwei Ausgangskanäle. Es kommen insgesamt nur acht Adressen zum Einsatz, da diese mehrmals verwendet werden können.
Für n Systeme werden nach diesem Konzept die Anzahl der Verbindungen A durch

$$A = (n - 1) * 4$$

berechnet.

Im Maximalfall können in einem „Parallel-Sysplex“ 32 Systeme beteiligt sein. In diesem Fall hat jedes System 31 benachbarte Systeme, mit denen es über vier Kanäle kommunizieren soll. Es werden also $31 * 4 = 124$ wiederverwendbare Adressen für die CTC Verbindungen unter den Systemen benötigt.
Im folgenden werden Auszüge aus den verschiedenen IEASYMxx Membern gezeigt, in denen die einzelnen Verbindungen zu den übrigen Systemen definiert werden. Im Falle von IEASYMS1 gibt es zwei Systeme, die mit einander kommunizieren sollen. Im Falle von IEASYMS9 werden drei Systeme über verschiedene Path-In (&PIxx =’yyyy’) und Path-Out (&POxx =’yyyy’) Definitionen mit einander verbunden.

Listing 6.8: CTC-Verbindungen bei 2 Systemen

```
/* ------------------------------------------------------------------ */
/* Member IEASYMS1: Define Static System Symbols                      */
/* ------------------------------------------------------------------ */
/* SYSNAME: EINS + VMUSERID(SYS1Z01)                                  */
/* SYSNAME: ZWEI + VMUSERID(SYS1Z02)                                  */
/* Beide mit &RACFSHR='NO'                                            */
/* ------------------------------------------------------------------ */
SYSDEF VMUSERID(SYS1Z01)
   SYSNAME(EINS)
   SYMDEF(&RACFSHR='NO')
   SYMDEF(&SSCPID='10')
   SYMDEF(&PI01='0E20')
   SYMDEF(&PI02='0E21')
   SYMDEF(&PO01='0E22')
```

```
    SYMDEF(&PO02='0E23')
SYSDEF VMUSERID(SYS1Z02)
    SYSNAME(ZWEI)
    SYMDEF(&RACFSHR='NO')
    SYMDEF(&SSCPID='11')
    SYMDEF(&PO01='0E40')
    SYMDEF(&PO02='0E41')
    SYMDEF(&PI01='0E42')
    SYMDEF(&PI02='0E43')
```

Listing 6.9: CTC-Verbindungen bei 3 Systemen

```
/* ------------------------------------------------------------------ */
/* Member IEASYMS9: Define Static System Symbols                      */
/* ------------------------------------------------------------------ */
/* SYSNAME: EINS + VMUSERID(SYS1Z01)                                  */
/* SYSNAME: ZWEI + VMUSERID(SYS1Z02)                                  */
/* SYSNAME: DREI + VMUSERID(SYS1Z03)                                  */
/* Alle mit &RACFSHR='NO'                                             */
/* ------------------------------------------------------------------ */
SYSDEF VMUSERID(SYS1Z01)
    SYSNAME(EINS)
    SYMDEF(&RACFSHR='NO')
    SYMDEF(&SSCPID='10')
    SYMDEF(&PI11='0E20')
    SYMDEF(&PI12='0E21')
    SYMDEF(&PO11='0E22')
    SYMDEF(&PO12='0E23')
    SYMDEF(&PI21='0E40')
    SYMDEF(&PI22='0E41')
    SYMDEF(&PO21='0E42')
    SYMDEF(&PO22='0E43')
SYSDEF VMUSERID(SYS1Z02)
    SYSNAME(ZWEI)
    SYMDEF(&RACFSHR='NO')
    SYMDEF(&SSCPID='11')
    SYMDEF(&PO11='0E40')
    SYMDEF(&PO12='0E41')
    SYMDEF(&PI11='0E42')
    SYMDEF(&PI12='0E43')
    SYMDEF(&PO21='0E20')
    SYMDEF(&PO22='0E21')
    SYMDEF(&PI21='0E22')
    SYMDEF(&PI22='0E23')
SYSDEF VMUSERID(SYS1Z03)
```

```
SYSNAME(DREI)
SYMDEF(&RACFSHR='NO')
SYMDEF(&PI11='0E20')
SYMDEF(&PI12='0E21')
SYMDEF(&PO11='0E22')
SYMDEF(&PO12='0E23')
SYMDEF(&PI21='0E40')
SYMDEF(&PI22='0E41')
SYMDEF(&PO21='0E42')
SYMDEF(&PO22='0E43')
```

Das folgende Bild (Abb.: 6.1) zeigt schematisch die CTC-Verbindungen zwischen drei Systemen.

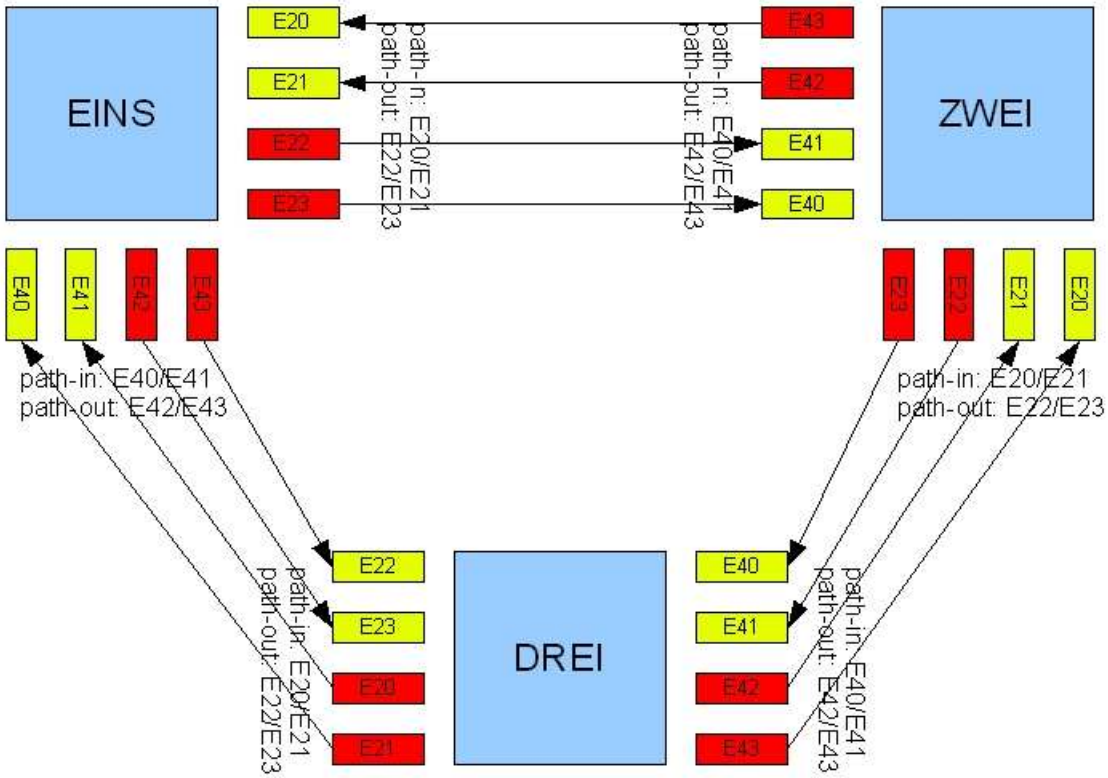

Abbildung 6.1: Schematische Darstellung der CTC-Verbindungen für drei Systeme

6.2.2 JES2PARM: JES für den MAS-Betrieb einrichten

Im Member `JES2PARM` des `SYS1.PARMLIB`-Datasets gibt es ein Statement, das für den Multi-Access-Spool zuständig ist. Hier werden alle Systeme eingetragen, die Zugriff auf den gemeinsamen Spoolbereich haben sollen. Sinnvollerweise sind dies alle Systeme, die zum Sysplex gehören. Es werden die Namen der Systeme eingetragen, die an dem globalen Spooling beteiligt sind. Ein drittes System müsste durch das ergänzen der Zeile

```
SID=SY03
```

in den MAS integriert werden.

7 Fazit / Ausblick

7.1 Fazit

Nachdem der Sysplex eingerichtet war und lief, kam eine Anfrage der IT-Akademie in Bayern aus Augsburg, einen weiteren Cluster bereit zu stellen, um einen praktisch orientierten Kurs mit dem Schwerpunkt WLM darauf abzuhalten. Als der zweite Cluster, der insgesamt fünf Systeme und zwei Coupling-Facilities beinhaltete, in Leipzig eingerichtet war, wurde klar, dass die Kommunikation der Systeme über die CF sehr viel Leistung beansprucht. Der Betrieb von einem Cluster mit zwei CFs und fünf Systemen, neben dem „normalen“ Betrieb, ist auf dem Rechner der Universität Leipzig mit sehr langen Antwortzeiten verbunden.
Der Betrieb von zwei oder mehr unabhängig laufenden Clustern parallel zueinander ist auf dem Rechner der Universität Leipzig wahrscheinlich nicht effizient zu realisieren, da nach dem Bauplan, der dieser Arbeit zugrunde liegt, pro Sysplex zwei CF benötigt werden.

Trotz der Antwortzeitenprobleme ist das hier beschriebene Verfahren des Klonens sehr effizient für Installationen, die viele identische Systeme nebeneinander betreiben. Durch das „sharen“ des größten Teils der Ressourcen wird der Platzbedarf der einzelnen Instanzen minimiert.
Durch die Minimierung des Platzbedarfs für neu angelegte Klone wird auch die Skalierbarkeit deutlich: Je Klon wird lediglich eine weitere Platte benötigt.

Das Erzeugen eines zusätzlichen Klons vom Master-Klon ist mit wenigen Schritten erledigt, was die Flexibilität stark erhöht.

Gerade für den Lehrbereich ist dieses Verfahren interessant, da auf diese Weise ohne großen Aufwand eine Lehrumgebung geschaffen werden kann, die den relativ geringen Anforderungen genügt. Zudem sind diese Systeme relativ gut an das Umfeld angepasst, in dem diese betrieben werden.

Beispielweise könnte im Bereich der Lehre jeder Student einer Praktikumsgruppe sein eigenes System betreiben, obwohl die Ressourcen in diesem Bereich eher knapp bemessen sind.

7.2 Ausblick

Das gesamte Verfahren des Klonens beruht auf der Tatsache, dass der größte Teil eines „s/390“- oder „z/OS“- Betriebssystems auf Platten abgelegt ist, auf die nur lesend zugegriffen wird. In einer Mainframe-Umgebung kann der lesende Zugriff auf ein und dieselbe Datei von mehr als einem Prozess erfolgen.
In einer Umgebung, die relativ begrenzt in Bezug auf CPU-Leistung, verfügbaren Arbeitsspeicher sowie verfügbaren externen Speicher ist, bietet das Klonen eine gute Möglichkeit, viele Systeme parallel miteinander zu betreiben. Jedoch stößt man leicht an die Grenzen der Umgebung. Im Folgenden werden zwei Aspekte betrachtet, die vor allem von der zur Verfügung stehenden Hardware beeinflusst werden.

7.2.1 UADS & BRODCAST global

In einer Standard-Installation sind die BRODCAST- und UADS-Datasets als lokale Ressourcen angelegt. Dies ist auch beim Masterklon und den von ihm abgeleiteten Klonen der Fall. Da die system-spezifischen Platten jedoch stark ausgelastet sind, wäre es hilfreich, Platz auf diesen Platten zu schaffen, indem das UADS- und BRODCAST-Dataset als globale Ressourcen anlegt werden. Voraussetzung dafür ist jedoch die Verknüpfung der Klone untereinander als Parallel-Sysplex , da die Datasets sonst für die Systeme nicht erreichbar sind.
Aus der Globalisierung dieser zwei Datasets ergeben sich neben dem vergrößerten Platz auf der system-spezifischen Platte außerdem noch zwei weitere Vorteile:

- In der Installation muss man nicht mehr pro System zwei Datasets pflegen und aktuell halten, bei vier Systemen also acht Datasets, sondern bei Globalisierung und maximal 32 Systemen insgesamt nur zwei.
- Um den Workload besser zu verteilen, kann sich jeder User an jedem System innerhalb des Sysplexes anmelden. Allerdings ist es durch die Globalisierung von UADS und BROADCAST- Datasets nicht mehr möglich, dass ein User seine Identifikation zu einem Zeitpunkt mehr als einmal innerhalb des Sysplexes verwendet.
  Ein Beispiel: Ein User schickt einen langlaufenden Job zu System „A“ , der über

Nacht laufen soll. Am nächsten Tag meldet er sich nicht an System „A“ an, sondern an System „B“. Die Nachricht, dass der Job auf System „A“ beendet wurde, wird er trotzdem erhalten, denn die Nachrichten werden über das globale BROADCAST-Dataset user-spezifisch zustellt unabhängig von dem System, an dem sich der User gerade angemeldet hat.

7.2.2 Spool lokal einrichten

In dem in diesem Kapitel beschriebenen Verfahren wird ein Multi Access Spool für alle Systeme eingerichtet. Der MAS stellt einen globalen Spoolbereich dar, der von allen Systemen genutzt wird. Jedoch ist es auch möglich, dass jedes System seinen eigenen Spoolbereich hat. Dies ist vermutlich in kommerziell genutzten Installationen nicht üblich, aber denkbar für Installationen aus dem Lehr-Bereich. Der MAS verbindet die Systeme auf eine sehr starke Art und Weise untereinander. Durch den MAS wird es ermöglicht, einen Job auf System „A“ abzuschicken und auf System „B“ ausführen zu lassen. Im Hinblick auf Lastverteilung ist dies ein wichtiger Aspekt, jedoch ist die Kapselung der Systeme voneinander ein gewünschter Effekt, beispielsweise in einer Praktikumsgruppe.
Ein Nachteil der lokalen Spool-Bereiche ist jedoch der größere Plattenbedarf. In der beschriebenen Konfiguration ist für einen Klon nur eine weitere Platte nötig. Diese Platte ist jedoch so stark ausgelastet, dass die Unterbringung des Spool-Bereiches eine zweite Platte erforderlich macht. Die lokalen Spool-Bereiche wären also mit einer Bedarfsteigerung von 100% verbunden.

7.2.3 Erweiterung der Klone

Eine Erweiterung der Klone mit IMS, DB2, CICS etc., um praxis-nahe Szenarien durchspielen zu können, wären Möglichkeiten für weitere Arbeiten.

Abkürzungen

Tabelle 7.1: Abkürzungen (Teil 1)

ACDS	Active-Control-Data-Set
ACS	Automated Control Storage
COMMDS	Communications-Data-Set
ARM	Automatic Restart Manager
APPN	Advanced Peer to Peer Networking
ASCII	American Standard Code for Information Interchange
BIOS	Basic Input Output System
BPAM	Basic Partitioned Access Method
BSAM	Basic Sequential Access Method
CDS	Couple Data Set
CF	Coupling Facility
CFR	Coupling-Facility-Receiver
CFS	Coupling-Facility-Sender
CFCC	Coupling Facility Control Code
CFRM	Coupling Facility Resource Manager
CICS	Customer Information Control System
COMMDS	Communications-Data-Set
CPU	Central Processing Unit
CSS	Channel Sub-System
CTC	Channel To Channel
DB2	Data Base 2
DASD	Direct Access Storage Device
DFSMS	Data-Facility-Storage-Management-Subsystem
DFDSS	Data-Facility-DataSet-Services
DOS	Disk Operating System (IBM)
DS	Data Set
EBCDIC	Extended Binary-Coded Decimal Interchange Code
ESA	Extended System Architecture
ESCON	Enterprise System CONnection
FICON	FIbre CONnection
FIFO	First-In-First-Out

Tabelle 7.2: Abkürzungen (Teil 2)

GRS	Global Resource Serialization
HAC	High-Availibility-Cluster
HFS	Hierarchical-File-System
HLQ	High-Level-Qualifier
HMC	Host-Managment-Console
HPCC	High Performance Computing Cluster
IML	Initial Microcode Load
IMS	Information Management System
IODF	Input Output Definition File
IP	Internet Protocol
IPL	Initial Program Load
IRLM	Inter-System Resource Lock Manager
ISPF	Interactive System Productivity Facility
JCL	Job Control Language
JES	Job Entry System
JES2	Job Entry System 2
JES3	Job Entry System 3
LASTCC	Last Condition Code
LIC	Licensed Internal Code
LBC	Load-Balancing-Cluster
LOGR	LOGR
LPAR	Logical Partition
MAS	Multiple Access Spool
MAXCC	Maximum Condition Code
MQSeries	Message Queueing System (asynchr. Transaktions-Monitor
MIF	Multiple-Image-Facility
MVS	Multiple Virtual Storage
NASA	National Aeronautics and Space Administration
NVSAM	Non-VSAM
OS	Operating System
OSA	Open Systems Adapter
PLPA	Pageable Link Pack Area
PR/SM	Processor Resource / Systems Manager
QSAM	Queued Sequential Access Method
RACF	Resource Access Control Facility
RLS	Record Level Sharing

Tabelle 7.3: Abkürzungen (Teil 3)

RMF	Resource Measurement Facility
RRS	Resource Recovery Services
SAP	System Assist Prozessor
SCDS	Source Control Data Set
SETI	Search for Extra-Terrestrial Intelligence
SFM	Sysplex-Failure-Management
SMF	System Management Facility
SMP	Symetrischer-Multi-Prozessor
SMS	System Managed Storage
SNA	Systems Network Architecture
SCDS	Source-Control-Data-Set
TCP	Transport Control Protocol
TPF	Transaction Processing Facility
TSO	Time Sharing Option
TSS	Time Sharing System
UADS	User Attribute DataSet
USS	Unix System Services
VM	Virtual Machine
VSAM	Virtual Storage Access Method
VSAM/RLS	VSAM Record Level Sharing
VSE	Virtual Storage Extended
VTAM	Virtual Telecommunication Access Method
VTOC	Volume Table Of Contents
VVDS	VSAM Volume Data Set
WLM	Work Load Manager
XCF	Cross System Coupling Facility
XML	eXtensible Markup Language

Literaturverzeichnis

[SysPl] W. Spruth, E. Rahm: Sysplex-Cluster-Technologien für Hochleistungs-Datenbanken, Datenbank-Spektrum, Heft 3/2002, S. 16-26.

[Parti] J. Buttlar,W. Spruth: Virtuelle Maschinen. zSeries und S/390 Partitionierung, IFE - Informatik Forschung und Entwicklung, Heft 1/2004, Juli 2004.

[inZos] Herrmann, Kebschull, Spruth: Einführung in z/OS und OS/390, 2. Auflage, Oldenburg Verlag, ISBN: 3-48627393-0.

[CFSiz] http://www-03.ibm.com/systems/z/cfsizer/

[HostL] Universität Leipzig, Institut für Informatik, Fachbereich Computersysteme: http://jedi.informatik.uni-leipzig.de/de/mainframe.html, 2008.

[sABC1] IBM International Technical Support Organization: ABCs of OS/390 System Programming Volume 1, IBM Form-Nr.: SG24-6981-01, 04/2008.

[sABC5] IBM International Technical Support Organization: ABCs of OS/390 System Programming Volume 5, IBM Form-Nr.: SG24-5655-00, 04/2000.

[zABC2] IBM International Technical Support Organization: ABCs of z/OS System Programming Volume 2, IBM-Form-Nr.: SG24-6982-02, 09/2008.

[zABC3] IBM International Technical Support Organization: ABCs of z/OS System Programming Volume 3, IBM-Form-Nr.: SG24-6983-02, 08/2007.

[zABC5] IBM International Technical Support Organization: ABCs of z/OS System Programming Volume 5, IBM-Form-Nr.: SG24-6985-01, 02/2008.

[zABC7] IBM International Technical Support Organization: ABCs of z/OS System Programming Volume 7, IBM Form-Nr.: SG24-6987-00, 08/2006.

[zABC9] IBM International Technical Support Organization: ABCs of z/OS System Programming Volume 9, IBM Form-Nr.: SG24-6989-03, 01/2008.

[zABC10] IBM International Technical Support Organization: ABCs of z/OS System Programming Volume 10, IBM Form-Nr.: SG24-6990-02, 08/2007.

[zABC11] IBM International Technical Support Organization: ABCs of z/OS System Programming Volume 11, IBM-Form-Nr.: SG24-6327-00, 11/2005.

[SPGLO] IBM International Technical Support Organization: System Programmer's Guide to: z/OS System Logger, IBM-Form-Nr.: SG24-6898-01, 07/2007.

[mGRSp] IBM International Technical Support Organization: MVS Planning: Global Resource Serialization, IBM-Form-Nr.: SA22-7600-06, 07/2007.

[JaMos] J. Moseley: VSAM Tutorial, http://www.jaymoseley.com/hercules/vstutor/vstutor.htm, 04/2008.

[UADS] IBM International Technical Support Organization: OSMVS Planning: Global Resource Serialization, IBM Form-Nr.: SA22-7600-06, 09/2007.

[gSchl] G. Schlüter: (Diplomarbeit) Untersuchungen zum CICS Transaction Gateway in der J2EE Umgebung, 2004.

[IBM 1] IBM International Technical Support Organization: Introduction to the New Mainframe: z/OS Basics, IBM Form-Nr. SG24-6366-00, 06/2006.

[DSN 1] IBM International Technical Support Organization: DFSMS Using Data Sets, IBM Form-Nr. SC26-7410-07, 09/2007

[BSI 1] Bundesamt für Sicherheit in der Informationstechnik: M 3.39 Einführung in die zSeries-Plattform, http://www.bsi.bund.de/gshb/deutsch/m/m03039.htm, 2008.

[BSI 2] Bundesamt für Sicherheit in der Informationstechnik: M 3.40 Einführung in das z/OS-Betriebssystem, http://www.bsi.bund.de/gshb/deutsch/m/m03040.htm, 2008.

8 Anhang

8.1 Inhalt der CD

8.1.1 PDF

Dieser Ordner enthält das Skript in Form einer pdf Datei.

8.1.2 Tex-Source

Dieser Ordner beinhaltet die Quell-Datein des Tex-Dokuments inklusive aller Bilder, die verwendet wurden.

8.1.3 Verwendete Jobs

ULISTOR.SYSPLEX.CNTL

Dieser Ordner beinhaltet alle Jobs, die ausgeführt wurden, um einen Masterklon eine Mini-z/OS-Instanz zu erstellen (Jobs für den Masterklon beginnen mit JOB0xx). Außerdem die Jobs, die ausgeführt worden sind, um einen weiteren Klon von diesem Masterklon zu erstellen (Diese Jobs beginnen mit JOB1xx).

ULISTOR.SYS1.SAMPLIB

Der Inhalt dieses Ordners spiegelt die Member des Datasets ULISTOR.SYS1.SAMPLIB wieder. Darin ist vor allem der Job enthalten, der benötigt wurde, um die Richtlinien für die CF zu dimensionieren.

8.1.4 Quellen

Alle verwendete Quellen.