



# HowTo: Installation von Oracle RAC 10g V.10.2.0.1 Enterprise Edition 64bit mit *redundanter* Shared Storage auf SuSE Linux Enterprise Server 9 SP2

Martin Klier <martin.klier@atu.de>  
Martin Blattner

Vorab-Version v. 17.01.2006.1

## 1 Einleitung

Dieses HowTo soll zur Erleichterung einer Installation des Oracle Real Application Clusters (RAC) der Version 10g / 10.2.0.1 auf SuSE Linux Enterprise Server 9 Service Pack 2 (SLES9 SP2) mit *redundanter* Shared Storage dienen.

Warum? Erstens mangelt es im deutschsprachigen Bereich an Anleitungen zum Thema, und zweitens möchten wir mit anderen DBAs ein paar Tricks teilen, die sich für die Umsetzung der Überschrift - zum Teil durch bittere Erfahrung - als notwendig erwiesen haben. Über Ihr Feedback freuen wir uns immer, Verbesserungs- und Ergänzungsvorschläge werden gerne gesehen, geprüft und ggf. mit aufgenommen.

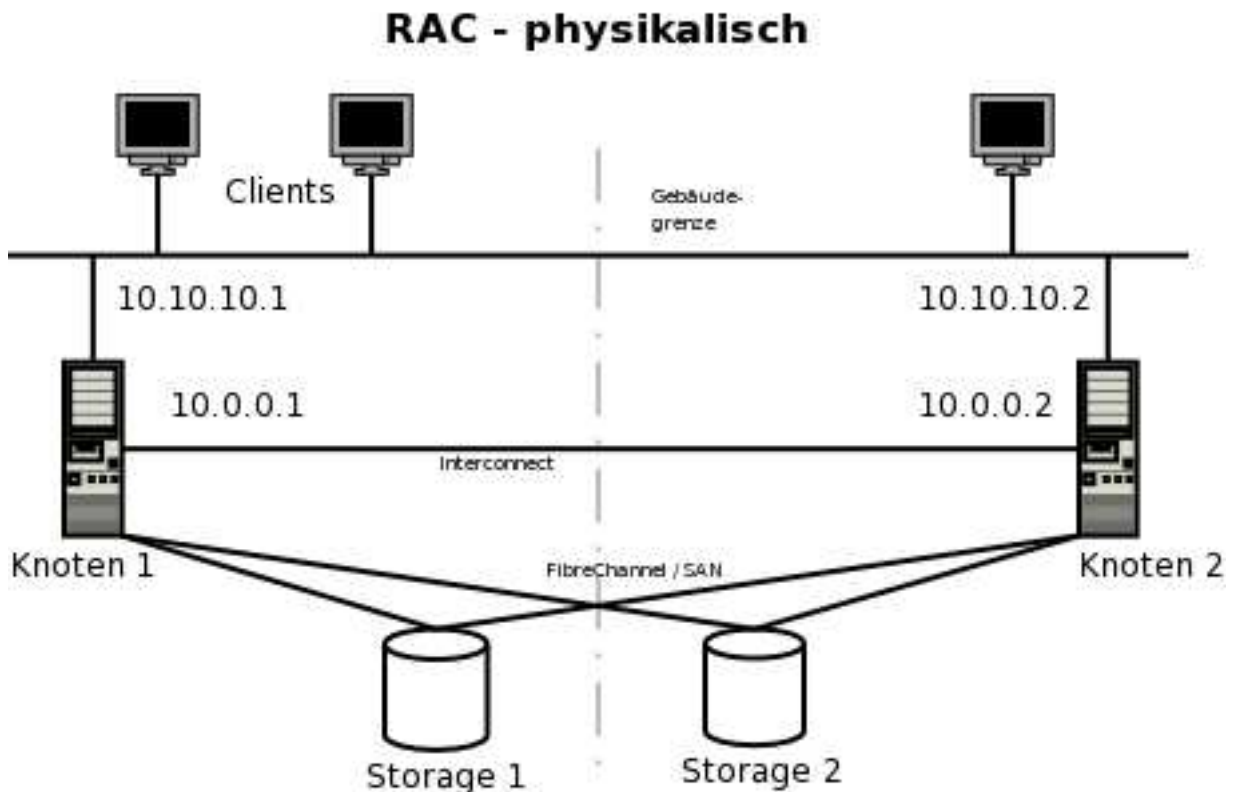
Wer? Die Adressaten sind Administratoren, die weniger Erfahrung im RAC- bzw. ASM-Umfeld haben und trotzdem vor der Notwendigkeit stehen, ein solches System zu implementieren. Allerdings sind gewisse Grundkenntnisse bezüglich Linux und Oracle Database notwendig.

### 1.1 Hinweis!

Wir bemühen uns um eine fachlich korrekte und für den Zweck erschöpfende Darstellung. Es wird jedoch keinerlei Haftung für entstehende Schäden, Unvollständigkeit oder Produktivitätsverluste etc. übernommen - weder durch A.T.U. noch durch die Autoren. Jeder Administrator ist selbst für Überprüfung und Test der implementierten Features sowie deren Zuverlässigkeit verantwortlich.

## 1.2 Ziel und Konzept

Abbildung 1: RAC - vereinfachter physikalischer Aufbau



Das Ziel ist die Herstellung eines RAC mit zu Redundanzzwecken softwareseitig verteilten Daten, die jedoch von beiden Knoten aus konkurrierend schreib-/lesbar sind (Mirrored Shared Storage). Abbildung 1 zeigt den physikalischen Aufbau.

Zur Verteilung der Daten im Storage-Pool nutzen wir ein „Bordmittel“ von Oracle 10g, das Automatic Storage Management (ASM). Dadurch ist keine (teure) storage-seitige Spiegelung über das SAN notwendig. Es handelt sich hierbei also um eine low-cost-Lösung, die jedoch durchwegs professionellen Ansprüchen genügt. Wir werden die Grundinstallation zwar kurz streifen, Schwerpunkte dieses Dokuments sind aber eindeutig Konfiguration und Administration der Oracle-Komponenten sowie ein Überblick über die nötigen theoretischen Zusammenhänge.

## 1.3 Umfeld

Die Anforderungen an ein Cluster sind üblicherweise:

- Hochverfügbarkeit (z.B. Produktivitätsverlust vermeiden)

- der Knoten
  - der Storages
  - der Infrastruktur
- räumliche Trennung (z.B. Absicherung gegen Verlust des RZ)

Die Hardwareplattform für unser HowTo besteht aus:

- 2 Knoten (z.B. HP ProLiant DL585 4xCPU AMD64, 8GB RAM)
- redundantes SAN in beiden RZ (z.B. 4x Brocade-Switch, FibreChannel)
- mehrere LUNs auf 2 Storages (z.B. EMC Clariion CX-300), symmetrisch angelegt
- redundanter Netzwerk-Interconnect mit 2x 1Gb Nettodurchsatz in privatem Netz / VLAN
- öffentliches Netz an beiden Knoten-Standorten

Wir gehen von diesen Softwarekomponenten aus:

- SuSE Linux Enterprise Server Version 9, Service Pack 2
- Das Oracle9i-Vorbereitungspaket von SuSE, orarun-1.8-109.5.i586.rpm
- Oracle Real Application Cluster 10g Release 2, Version 10.2.0.1
- Oracle Database 10g Enterprise Edition, Version 10.2.0.1

## 2 Vorbereitung

### 2.1 Global

Der SLES9 64bit mit SP2 ist eine von Oracle für Version 10g 64bit Enterprise Edition zertifizierte Basis. Voraussetzung für Installation und Betrieb ist die fachgerechte Montage der Systeme und der richtige Anschluß aller Verbindungen. Für ein später zuverlässiges Cluster sollten alle Formen der Redundanz (lokale HDDs, Spannungsversorgung / Netzteile, Netzwerkkomponenten etc.) schon vor Beginn der Installation (bei der Erstinbetriebnahme der Geräte) getestet werden. Die sprichwörtlichen kleinen Ursachen werden sonst ziemlich sicher das gesamte „hochverfügbare“ Cluster ad absurdum führen.

Unter Umständen kann es sinnvoll sein, einen Knoten vorzukonfigurieren und dann zu klonen. Nutzer, die mit dem NaviAgent-Client und einer EMC Clariion arbeiten möchten, können wir die Klon-Methode jedoch nicht empfehlen. In unserer Kombination war ein Einbinden des geklonten Systems in die Storagegroup der EMC nicht möglich. Die nun gelisteten Punkte gelten für beide Knoten des Clusters.

### 2.1.1 Pakete

Wählen Sie Ihre Softwarepakete sinnvoll. Niemand hat etwas von aufgeblähten Serverinstallationen, doch ist auf der anderen Seite ständiges Nachinstallieren mindestens lästig und eventuell sogar riskant. Oracle setzt zwar eine graphische Umgebung zur Installation voraus, es ist jedoch ohne weiteres möglich für die Einrichtung per *ssh -X* z.B. vom Notebook aus ohne lokalen X-Server zu arbeiten. Prüfen Sie deshalb, ob Sie auf dem Serversystem unbedingt eine graphische Umgebung benötigen.

1. Zur Nutzung von heruntergeladenen Oracle-Installationsquellen benötigen Sie *cpio* auf dem Knoten, der später die Installation durchführen soll.
2. Installieren Sie außerdem das *orarun.rpm* für die Database 9i-Versionen, es ist mit einigen Änderungen sehr brauchbar.
  - (a) Ermöglichen Sie dem User oracle die Anmeldung im System durch Vergabe einer Login-Shell und eines Passworts, machen Sie oracle zum Mitglied der Gruppe dba und testen Sie den Zugang. Führen Sie zur Prüfung *id* aus.
  - (b) Editieren Sie */etc/sysconfig/oracle* und verhindern Sie den Start der diversen Services durch den Parameter "no", beispielsweise `START_ORACLE_DB="no"`. Einen Vorschlag ohne Kommentarzeilen sehen Sie unter Algorithm 1 auf der nächsten Seite. Diese verschiedenen Aufgaben übernimmt später der CRS-Dienst. Wird hier nicht deaktiviert, treten später beim Booten des Systems allerlei seltsame Seiteneffekte auf. Auch das Setzen der Kernelparameter erfolgt an dieser Stelle. Mindestens SHMMAX sollte auf einen sinnvollen Wert angepasst werden. In unserem Beispiel blieb der Einfachheit halber der durch das *orarun.rpm* definierte Defaultwert stehen.

**Algorithm 1** /etc/sysconfig/oracle

---

```

ORACLE_OWNER="oracle"
ORACLE_BASE=/opt/oracle
START_ORACLE_DB="no"
START_ORACLE_DB_LISTENER="no"
START_ORACLE_DB_CONSOLE="no"
START_ORACLE_DB_AGENT="no"
START_ORACLE_DB_APACHE="no"
ORACLE_DB_APACHE_USE_SSL="no"
START_ORACLE_DB_CMANAGER="no"
START_ORACLE_DB_OID="no"
START_ORACLE_RAC_OCFS="no"
START_ORACLE_RAC_OCM="no"
ORACLE_RAC_OCM_PARAMETERS=""
START_ORACLE_RAC_GSD="no"
SET_ORACLE_KERNEL_PARAMETERS="yes"
SHMMAX=3294967296
SHMMNI=4096
SHMALL=2097152
SHM_USE_BIGPAGES=0
SEMMSL=1250
SEMMNS=32000
SEMOPM=100
SEMMNI=256
IP_LOCAL_PORT_RANGE="1024 65000"
RMEM_MAX=262144
RMEM_DEFAULT=262144
WMEM_MAX=262144
WMEM_DEFAULT=262144
FILE_MAX_KERNEL=131072
FILE_MAX_SHELL=65536
PROCESSES_MAX_SHELL=16384
MAX_CORE_FILE_SIZE_SHELL=unlimited
VM_MAPPED_RATIO=100
AIO_MAX_SIZE=262144

```

---

- (c) Passen Sie die */etc/profile.d/oracle.sh* an, ein Beispiel mit empfohlenen Werten sehen Sie unter Algorithm 2 auf der nächsten Seite.
- i. Ändern Sie die erste Zeile so, daß das Skript für alle User des Systems oder mindestens noch für root durchlaufen wird.
  - ii. Passen Sie die Variable `ORACLE_SID` an: Auf node1 TESTR1, auf node2 TESTR2!
  - iii. Ändern Sie die Variable `ORACLE_HOME` auf die Endung `/10g/db`.
  - iv. Fügen Sie die Variable `ORA_CRS_HOME` ein, sie soll auf

- \$ORACLE\_BASE/crs zeigen.
- v. Ändern Sie NLS\_LANG und NLS\_SORT für Ihre Zwecke ab.
  - vi. Kommentieren Sie außerdem die letzte Zeile mit LD\_ASSUME\_KERNEL aus und/oder ersetzen Sie sie wie gezeigt.

---

**Algorithm 2** oracle.sh
 

---

```

if true; then #[ 'id -un' == "oracle" ]; then
if test -f /etc/sysconfig/oracle; then
. /etc/sysconfig/oracle
else
if test -f /etc/rc.config.d/oracle.rc.config; then
. /etc/rc.config.d/oracle.rc.config
else
if test -f /etc/rc.config; then
. /etc/rc.config
fi
fi
fi
ORACLE_SID=TESTR1 ##### auf 2. Knoten: TESTR2
ORACLE_BASE=/opt/oracle
ORACLE_HOME=$ORACLE_BASE/product/10g/db
ORA_CRS_HOME=$ORACLE_BASE/crs
# ...
# ...
NLS_LANG=GERMAN_GERMANY.WE8MSWIN1252
export NLS_LANG
NLS_SORT=BINARY
export NLS_SORT
unset JAVA_BINDIR JAVA_HOME
ulimit -c ${MAX_CORE_FILE_SIZE_SHELL:-0} 2>/dev/null
ulimit -u ${PROCESSES_MAX_SHELL:-16384} 2>/dev/null
ulimit -n ${FILE_MAX_SHELL:-65536} 2>/dev/null
# test -d /lib/i686 && export LD_ASSUME_KERNEL=2.2.5
export LD_ASSUME_KERNEL=2.4.21
fi

```

---

3. Installieren Sie die Hersteller-Support-Pakete Ihrer Hardware (z.B. HP ProLiant Support Pack), es liegen hier oft optimierte Treiber, Konfigurationen, Fernwartungs- und Diagnoseprogramme bei.

### 2.1.2 Kernel

Patchen Sie den Kernel möglichst früh auf die für Oracle 10gR2 mindestens erforderliche Version 2.6.5-7.201 (oder natürlich besser) und denken Sie bei der Gelegenheit auch an die Kernel Sources. SuSE bietet für das Kernelupgrade drei Pakete an: kernel-smp/-default, kernel-source und kernel-syms für die Symboltabellen.

Um später beim Start der Datenbankinstanz den Fehler „ORA-27125: unable to create shared memory segment“ zu vermeiden, befolgen Sie Note 293988.1 wie im folgenden beschrieben. Der Kern des Problems ist, daß der Kernel das Privileg „MLOCK“ für hugetlb-memory benötigt. Durch

```
echo 1 > /proc/sys/vm/disable_cap_mlock
```

sollte das Problem temporär gelöst sein. Um die Sache persistent zu machen, tragen Sie in die */etc/sysctl.conf* die Zeile

```
vm.disable_cap_mlock = 1
```

ein und führen Sie

```
/etc/init.d/boot.sysctl start
```

aus. Danach muß dieses Bootscript noch in die Runlevelkonfiguration aufgenommen werden. Für SuSE gilt hier:

```
chkconfig boot.sysctl on
```

### 2.1.3 Zeichensatz

Nichts ist ärgerlicher als ein falscher Zeichensatz! Konfigurieren Sie deshalb im YaST Sysconfig-Editor *RC\_LANG* unter System▷Environment▷Language Ihren Anforderungen entsprechend, z.B.

```
de_DE@euro
```

## 2.2 Netzwerk

### 2.2.1 Öffentliches Netz

Hier ist in jedem Fall 100Mbit-Hardware wünschenswert. Stellen Sie auf beiden Knoten im YaST Ihre gewünschte IP-Adresse für das öffentliche Netz ein.

### 2.2.2 Interconnect mit NIC-bonding

Ein Cluster-Interconnect mit weniger Bandbreite als 1000Mbit/s dürfte für produktive Datenbanken in jedem Fall zu wenig sein. Halten Sie sich hier nicht kurz, die Performance der CacheFusion-Technologie und damit des ganzen RAC steigt und fällt mit der hier möglichen Übertragungsrate.

(Sollten Sie auf die ggf. verdoppelte Bandbreite und die Sicherheit eines redundanten Interconnects verzichten wollen, vergeben Sie lediglich pro Knoten eine „private“ IP-Adresse und lesen bitte unter 2.2.3 auf der nächsten Seite weiter.)

Gehen Sie für die Einrichtung der gebündelten Netzwerkkarten für den Interconnect auf beiden Knoten analog vor: Richten Sie beide Adapter mit YaST ein, die erste mit der später für das Bondinginterface *bond0* gewünschten IP-Adresse, die zweite mit einer beliebigen Dummy-IP. Damit erzeugen Sie provisorisch Konfigurationsdateien unter

*/etc/sysconfig/network* nach dem Muster *ifcfg-eth-id-<mac-address>*.

Extrahieren Sie nun aus beiden Dateien die PCI-Bus-ID des Adapters, die ähnlich „bus-pci-0000:03:08.0“ aussehen wird. Kopieren Sie nun die *ifcfg-eth-id*-Datei mit der korrekt gewählten IP-Adresse auf *ifcfg-bond0* und löschen Sie die beiden provisorisch erzeugten Konfigurationsfiles.

Editieren Sie *ifcfg-bond0* und fügen Sie die in Algorithm 3 gezeigten drei ersten Zeilen mit Ihren Werten versehen ein. Durch diese Art der Konfiguration kennt der Bondingtreiber nativ die Abhängigkeiten seiner zu „versklavenden“ Netzwerkadapter. Bitte nach etwaigen späteren Umbauten auf dem PCI-Bus prüfen, ob die IDs noch passen, der Interconnect ist lebenswichtig für das Cluster!

---

#### Algorithm 3 ifcfg-bond0

---

```
BONDING_MASTER=yes
BONDING_SLAVE_0='bus-pci-0000:08:0c.0'
BONDING_SLAVE_1='bus-pci-0000:03:08.0'
BOOTPROTO='static'
BROADCAST='10.0.0.255'
IPADDR='10.0.0.1'
MTU=""
NETMASK='255.255.255.0'
NETWORK='10.0.0.0'
REMOTE_IPADDR=""
STARTMODE='onboot'
UNIQUE='Uog3.qZvrV0AxcUD'
_nm_name='bus-pci-0000:03:08.0'
```

---

Für die genaue Konfiguration des Bondings empfehlen wir das sehr gute „*Linux Ethernet Bonding Driver mini-howto*“, das Sie im System unter */usr/src/linux/Documentation/networking/bonding.txt* finden.

**Tip:** Der Modus „balance-xor“ hat sich für uns bestens bewährt. Konfigurieren Sie außerdem insbesondere die diversen `arp_*`- und `mii*`- Parameter mit besonderer Umsicht. Denn nur dann erreichen Sie wirkliche Ausfallsicherheit.

Nun tragen Sie Ihre gewünschte Modul-Konfiguration in `/etc/modprobe.conf.local` ein. Zur Herstellung der Bootfähigkeit kann u.U. sinnvoll sein, daß Sie mit einem entsprechenden Eintrag im YaST Sysconfig-Editor unter `System > Kernel > MODULES_LOADED_ON_BOOT` das Modul „bonding“ schon früh in den Kernel laden.

Ein `rcnetwork restart` sollte nun die gebündelten Netzwerkkarten pro Knoten unter gemeinsamer IP- und MAC-Adresse erscheinen lassen. Testen Sie die Verbindung auch mit mehreren parallelen `ping -fs65500 $IP`, um einen ersten Eindruck vom Lastverhalten zu erlangen. Das Tool `gkrellm` (leider bei SLES9 nicht dabei - das RPM-Paket von SuSE Linux Professional 9.1 sollte jedoch funktionieren) leistet hier zur Überwachung gute Dienste. Führen Sie auch Versuche zum Load Balancing und Fail-Over-Verhalten durch.

**Anmerkung:** Nicht alle Switches unterstützen alle möglichen bonding-Modi. Besonders bei einem Switch-Verband können doppelte MAC-Adressen ernsthafte Probleme bereiten. Hier sind wieder individuelle Tests gefordert. Für unser Cluster hat sich der Einsatz von zwei eigenen Glasfaserstrecken mit vier 1000Mbps-Monomode-Medienkonvertern von Kupfer auf Glas bewährt: Diese de facto Cross-Over-Kabel vereinen Flexibilität und Ausfallsicherheit in besonderem Maße.

### 2.2.3 /etc/hosts

Machen Sie die verschiedenen Systeme einander unabhängig vom DNS-System durch Namen in der `/etc/hosts` bekannt. Es ist ja möglich, daß es für Ihren Interconnect gar keine eingetragenen DNS-Namen geben kann - hier ist die „klassische“ Version fast immer erste Wahl. Algorithm 4 erklärt sich selbst.

---

#### Algorithm 4 /etc/hosts

---

```
# Auszug aus /etc/hosts beider Knoten
# Knoten 1
10.10.10.1 node1.example.com node1 # Knoten 1
10.10.10.11 node1-vip.example.com node1-vip # virtuelle IP
10.0.0.1 node1-int.example.com node1-int # Interconnect
# Knoten 2
10.10.10.2 node2.example.com node2 # Knoten 2
10.10.10.22 node2-vip.example.com node2-vip # virtuelle IP
10.0.0.2 node2-int.example.com node2-int # Interconnect
# ggf. sind zusätzlich die Serviceprozessoren
# Ihrer Storage einzutragen.
```

---

Leider hat YaST die unangenehme Eigenschaft, die Datei `/etc/hosts` „aufräumen“ und „verschönern“ zu wollen. Dann erfolgt die Eintragung wild zusammengesetzter IP-Adress- und Namenskombinationen am Ende der Datei. Geschieht das im ungeeigneten Augenblick, ist es später nicht mehr möglich die ASM-Instanz zu starten. Diese beklagt sich dann u.U. über „*ORA-27504: IPC error creating OSD context*“ und ähnliche Dinge, die nur schwer auf den eigentlichen Unruhestifter zurückzuführen sind. Fügen Sie deshalb die beiden Parameter `CHECK_ETC_HOSTS` und `BEAUTIFY_ETC_HOSTS` in die `/etc/sysconfig/suseconfig` ein, und setzen Sie dafür den Wert „no“. Algorithm 5 zeigt den Kopf der Datei.

---

**Algorithm 5** `/etc/sysconfig/suseconfig`


---

```
CHECK_ETC_HOSTS="no"
BEAUTIFY_ETC_HOSTS="no"
## Path: System/SuSEconfig
## Type: yesno ## Default: yes
# .....
```

---

### 2.2.4 ssh, Schlüsseltausch

Der Oracle-Cluster-Dienst benötigt für Installation und Betrieb einen promptfreien Fernzugriff auf alle Knoten des künftigen Clusters. Oracle unterstützt die Verwendung von OpenSSH, das auf die Akzeptanz von zuvor ausgetauschten Schlüsseln konfiguriert werden muß.

Bei erhöhtem Sicherheitsbedarf sollte die Verwendung eines `ssh-agent` zur Passwortverwaltung der Schlüssel erwogen werden. Dann sind jedoch hier nicht besprochene Vorkehrungen beim Booten der Systeme zu treffen.

Erzeugen Sie mit `ssh-keygen -b2048 -trsa` einen Key für die Benutzer `root` und `oracle` auf jedem der Knoten, und übertragen Sie sie mit `ssh-copy-id -i ~/.ssh/id_rsa.pub $ZIEL` an alle möglichen `user@interface` - Kombinationen. Das heißt:

- an das jeweils andere Node - öffentliches Netz und Interconnect
- an die lokale Maschine - öffentliches Netz und Interconnect  
Wichtig! - das Installationsprogramm des CRS greift später auch auf den (lokalen) Knoten, der die Installation durchführt, über ssh zu!!

Testen Sie *unbedingt* alle Kombinationen, da `ssh` bei unbekanntem fremden Schlüsseln zurückfragt, ob dieser in die `~/.ssh/known_hosts` aufgenommen werden darf. Es darf zur Installation jedoch *keine interaktive Eingabe* mehr nötig sein.

## 2.3 Volumes

### 2.3.1 Hardwarezugriff

Stellen Sie sicher, daß Ihre FibreChannel-Adapter korrekt konfiguriert sind, und die LUNs dem System als Blockgerät zur Verfügung stehen. Die nötigen Arbeitsschritte hier zu erläutern wäre zu weitführend, und außerdem für die wenigsten Leser brauchbar.

**Anmerkung:** Es kann aus Gründen des Setups (z.B. für die Kombination EMC und QLogic-Adapter) sinnvoll sein, die LUNs zu partitionieren und später „raw“ auf Partitionen statt direkt auf Devices zuzugreifen.

### 2.3.2 Benötigte Volumes

Der Zugriff von ASM und Cluster-Dienst auf die physikalischen Geräte findet über Rawdevices statt. Es ist seit Oracle RAC 10g Release 2 möglich, OCR und Voting Disk durch Oracle automatisch zu spiegeln. Die Anzahl der OCRs *muß gerade*, die der Voting Disks (Quoren) *muß ungerade* sein. Es ist daher sinnvoll, auf einer eventuell vorhandenen „billigen“ dritten Storage ein wenig Platz für das dritte Quorum freizuhalten. Ob diese Storage dann wie unter Punkt 2.3.3 auf der nächsten Seite über Multipath angesprochen werden kann, muß im Einzelfall geprüft werden.

Sie benötigen mindestens die folgenden physikalischen Volumes/LUNs:

- Storage 1:
  - 100MB für die Voting-Disk / Quorum 1
  - 100MB für die Oracle Cluster registry (OCR) 1
  - Nutzdatenvolume 1
  
- Storage 2:
  - 100MB Quorum 1
  - 100MB OCR 2
  - Nutzdatenvolume 2
  
- Storage 3
  - 100MB Quorum

### 2.3.3 Multipath

Für mehrere Kombinationen aus HBAs und Storage-Systemen (in unserem Falle Qlogic und EMC CX-300) existiert die Unterstützung von Mehrpfadfähigkeit. D.h. in einer SAN-Umgebung, die theoretisch mehrere Möglichkeiten bietet auf die Storage(s) zuzugreifen, wird unter allen möglichen Pfaden ein verfügbarer herausgegriffen und verwendet. Auch Loadbalancing zur gleichmäßigen Auslastung der Infrastruktur ist denkbar. Die Herstellung eines einfachen Multipath-Setups erläutern wir Ihnen hier schrittweise:

1. Die möglichen Pfade zur LUN treten auf anfangs recht lästige Weise auf den Plan: Im Verzeichnis `/dev/` erscheint nach dem Laden des FC-HBA-Treibers für jeden möglichen Pfad ein eigenes SCSI-Gerät `sd*`. Welches ist nun korrekt? Grundsätzlich einmal jedes. Aber um die LUNs aus dem Gewirr von Geräten (man denke an unsere Ausgangslage: zwei Storage-Systeme, diverse LUNs wie im Folgenden noch erläutert, jede LUN zur Spiegelung auf beiden Storages) wieder herauszufinden, identifiziert man seine LUNs eindeutig durch ihre world wide numbers (WWN, wird durch das Storage-System vergeben und gilt für die gesamte Lebensdauer der LUN). Für uns erledigt diese Arbeit das Paket *Multipath*, das für SLES9 mitgeliefert wird. Es schafft sozusagen einen Alias mit unveränderlichem Namen für zwei oder mehrere (ggf. wechselnde) `sd*`-Geräte. Installieren Sie *Multipath* mit YaST.
2. Viele Storage-Systeme erfordern eine Anpassung an das verwendete Zugriffsverfahren. Im Fall der EMC CX-300 muß dazu ein Parameter auf der Storage umgestellt werden:  
failover mode = 1 für „PowerPath“.
3. Die Treiber für den FibreChannel HBA müssen für das Multipath-Setup zertifiziert sein, Informationen über die benötigte Version finden Sie bei Ihrem Storage- oder HBA-Hersteller. Prüfen Sie, ob die von Ihrem SLES9 oder Ihren hardware-spezifischen Treibersets mitgelieferten HBA-Treiber den Anforderungen genügen oder ob Sie selbst eine aktuelle Version installieren müssen. Überprüfen Sie nach einer etwaigen Installation bitte, ob `/etc/modprobe.conf.local` noch Ihren Wünschen entspricht, bevor Sie fortfahren. In unserem Falle hatte jedoch das HP ProLiant Support Pack schon gut vorgesorgt und ohne unerwünschte Nebenwirkungen die erforderliche Version für unsere QLA2340 HBAs mitgebracht.
4. Tragen Sie nun mit dem YaST Sysconfig-Editor unter System▷Kernel das HBA-Modul in die Parameter `INITRD_MODULES` und `MODULES_LOADED_ON_BOOT` ein, um dessen Verfügbarkeit in jedem Fall sicherzustellen. YaST erzeugt Ihnen danach eine neue `initrd`. Bitte beachten Sie auch, daß nicht Modul gleich Modul ist: In unserem Falle wäre „QLA2300“ das benötigte Modul, jedoch muß „QLA2xxx“ in die obengenannten Parameter eingetragen werden, damit beim Laden alle Konfigurationen eingelesen werden.

5. Laden Sie nun testweise das multipath-Modul mit `modprobe dm_multipath` und sehen Sie sich Ihre Multipath-Umgebung mit `/sbin/multipath -d` an. (Der Parameter `-d` bewirkt einen dry-run, d.h. bei diesem ersten Aufruf wird noch nichts verbunden. Möchten Sie bereits verbundene Pfade betrachten, nutzen Sie stattdessen `-ll`.) Sind Sie mit dem Ergebnis zufrieden, spricht nichts mehr gegen den Aufruf von `/sbin/multipath` ohne Parameter. Damit wird das Verzeichnis `/dev/disk` erzeugt, und unter `/dev/disk/by-name` für jede LUN *und* für jede Partition ein eigenes Gerät angelegt. Beachten Sie, daß die LUNs hier nicht partitioniert werden können. Partitionen müssen über die korrespondierenden `/dev/sd*`-Geräte angelegt, die HBAs zum Neueinlesen der Geräte gebracht und `/sbin/multipath` erneut angerufen werden.
6. Hat dies alles funktioniert, fügen Sie nun mit `chkconfig -a boot.multipath` das initiale Laden des Moduls und den Erstaufruf von Multipath ein. Auch sollte überprüft werden, ob die spätere automatische Pfadüberwachung durch den `multipathd` aktiviert ist, ansonsten auch diesen mit `inserv multipathd` einfügen. Jedoch müssen Sie am Ende der Gesamtinstallation ohnehin alle Services auf ihre Reihenfolge hin überprüfen um ein rebootsicheres System herzustellen.

### 2.3.4 RAW-Devices

Ist alles so weit in Ordnung, erhalten Sie einen Zwischenstand wie:

```
node1:/dev/disk/by-name # ll
total 8
drwxr-xr-x  2 root root 4096 Dec 27 11:49 .
drwxr-xr-x  5 root root 4096 Dec 16 15:09 ..
<...> 36006019fd0e98db0ad911 -> ../../dm-6
<...> 36006019fd0e98db0ad911p1 -> ../../dm-23
<...> 3600601afd0e98db0ad911 -> ../../dm-5
<...> 3600601afd0e98db0ad911p1 -> ../../dm-31
<...> 3600601bfd0e98db0ad911 -> ../../dm-4
<...> 3600601bfd0e98db0ad911p1 -> ../../dm-24
<...> 3600602a73bc9edb0ad911 -> ../../dm-3
<...> 3600602a73bc9edb0ad911p1 -> ../../dm-19
<...> 3600603073bc9edb0ad911 -> ../../dm-0
<...> 3600603073bc9edb0ad911p1 -> ../../dm-17
```

Hier stellt jeweils der erste Link die LUN und der zweite dessen erste Partition (\*p1) dar. Konfigurieren Sie nun in `/etc/raw` die Bindung zwischen Rawdevice und Blockdevice/Partition in der Form `rawdevice:blockdevice` (z.B. „raw1:36006019fd0e98db0ad911p1“) und vergeben Sie im Verzeichnis `/dev/raw/` für alle benutzten Rawdevices Berechtigung 660 für Benutzer oracle und Gruppe oinstall:

```
ls -l /dev/raw/raw{1,2,3,4,5,6}
chmod 660 /dev/raw/raw{1,2,3,4,5,6}
chown oracle:oinstall /dev/raw/raw{1,2,3,4,5,6}
ls -l /dev/raw/raw{1,2,3,4,5,6}
```

Nun aktivieren Sie die Änderungen mit `rcraw stop`; `rcraw start`, und prüfen mit `/usr/sbin/raw -qa` den Erfolg. Das Ergebnis sollte in etwa lauten:

```
node1:~ # /usr/sbin/raw -qa
/dev/raw/raw1: bound to major 8, minor 81
/dev/raw/raw2: bound to major 8, minor 33
/dev/raw/raw3: bound to major 8, minor 49
/dev/raw/raw4: bound to major 8, minor 65
/dev/raw/raw5: bound to major 8, minor 1
/dev/raw/raw6: bound to major 8, minor 17
```

Um den Überblick zu bewahren, empfehlen wir eine ausführliche Dokumentation innerhalb der `/etc/raw`. Sie kann übrigens dank Multipath auf allen Knoten identisch sein, da ja alle LUNs für alle Knoten sichtbar sind und ihre Identifier nur von der Storage abhängen. Dies schafft eine nicht zu verachtende Übersichtlichkeit für den Administrator. Algorithm 6 zeigt ein Beispiel für die Datei `/etc/raw`.

---

**Algorithm 6** `/etc/raw`


---

```
# /etc/raw
# Modifiziert für multipath-Devices by-name
#
# Quorum 1 EMC 1 LUN65
raw1:disk/by-name/360060770add31e40ad911p1
# Quorum 2 EMC 2 LUN63
raw2:disk/by-name/360060750add31e40ad911p1
# Quorum 3 EMC 3 LUN90
raw3:disk/by-name/3600603073bc9edb0ad911p1
# OCR 1 EMC 1 LUN60
raw4:disk/by-name/360060720add31e40ad911p1
# OCR 2 EMC 2 LUN70
raw5:disk/by-name/3600604a1937c0da0ad911p1
# RACVOL 1 EMC 1 LUN EMC_1_AWIS_01_180GB
raw6:disk/by-name/36006003cff8269227bd911p1
# RACVOL 2 EMC 2 LUN EMC_2_AWIS_01_180GB
raw7:disk/by-name/36006056cc0795217bd911p1
# AUXVOL 1 EMC 1 LUN EMC_1_AWIS_02_140GB
raw8:disk/by-name/360060843fc28d227bd911p1
# AUXVOL 2 EMC 2 LUN EMC_2_AWIS_02_140GB
raw9:disk/by-name/360060a08c1ec9217bd911p1
```

---

Stellen Sie nun die Bootreihenfolge über die Manipulation der *insserv*-Header (Hinzufügen der Abhängigkeiten) aller beteiligten Services ein:

1. boot.multipath
2. multipathd
3. raw

und führen Sie für jeden Dienst

`/sbin/insserv <Service>` aus. Algorithm 7 auf Seite 19 zeigt anhand des `init.crs` ein Beispiel für den Header.

Beobachten Sie bei gestartetem *multipathd* wie sich die Ausgabe von *multipath -ll* ändert, wenn Sie abwechselnd das eine oder andere FibreChannel-Kabel von den Knoten abziehen.

## 2.4 Abschluß

Auch wenn es dem Linux-Administrator widerstrebt: Rebooten Sie hier die Systeme, damit wirklich alle Prozesse von der neuen Umgebung profitieren. Stellen Sie sicher daß die volle Funktionalität auch nach dem Reboot auf beiden Maschinen wieder vorhanden ist. Sie wissen schon: Anmelden als root und oracle, *ping*, *raw -qa*, *ssh*, *ifconfig* und so weiter. Auch ein erneutes *multipath -ll* zur Sicherheit kann nicht schaden, denn jetzt geht es ans Eingemachte.

# 3 Cluster-Ready-Services CRS

## 3.1 Installation

Melden Sie sich als oracle an und erstellen Sie ein Verzeichnis für die Installationsquellen. Vorschlag: `/opt/oracle/install`.

Zuallererst benötigen Sie eine ausreichende Oracle-Lizenz und einen geeigneten Installationssatz für 64bit CRS. Verwenden Sie z.B. *ship.crs.lnxx86-64.cpio.gz* für Linux x86-64bit von Oracle TechNet. Entpacken und de-archivieren Sie die Datei in einem eigenen Verzeichnis, etwa `/opt/oracle/install/crs`, mit

```
gunzip ship.crs.lnxx86-64.cpio.gz
cpio -idmv < ship.crs.lnxx86-64.cpio
```

- es entsteht ein Unterverzeichnis `./Disk1` mit allen Installationsdaten.

1. Wechseln Sie als Benutzer Oracle in das eben erzeugte Verzeichnis `./Disk1` und starten Sie das Script `./runInstaller` mit gültiger `$DISPLAY`-Variable Ihrer Shell.

2. Wichtig: Vergeben Sie für OUIhome1 ein anderes Verzeichnis als das später für das RDBMS geplante. Wählen Sie den Wert von \$ORA\_CRS\_HOME, bei uns beispielsweise  
*/opt/oracle/product/10g/crs*
3. Vergeben Sie einen sprechenden Namen für das Cluster an sich.  
Beispiel: TESTCRS
4. Tragen Sie die Public- und Private Node Names wie schon unter Punkt 2.2.3 auf Seite 9 spezifiziert ein, siehe Tabelle 1.

Tabelle 1: CRS Node Names

Public Node Name	Private Node Name	Virtual Node Name
node1.example.com	node1-int.example.com	node1-vip.example.com
node2.example.com	node2-int.example.com	node2-vip.example.com

5. Geben Sie die Verwendung der Netzwerkschnittstellen an. Hier werden - abhängig von Ihrer gewählten Konfiguration und der Anzahl der verbauten/erkannten Adapter - mehrere Interfaces vorgeschlagen. Tabelle 2 auf der nächsten Seite zeigt den Fall für Bonding ohne weitere Netzwerkkarten.

Tabelle 2: CRS Netzwerkschnittstellen

Interface Name	Subnet	Interface Type	Bemerkung
bond0	10.0.0.0	Private	-
eth0	10.10.10.0	Public	-
eth2	10.0.0.0		(versklavt)
eth4	10.0.0.0		(versklavt)

**Vorsicht:** Sie sehen die Interfaces, die sie zu einem bond zusammengefügt haben, weiterhin als einzelne Adapter aufgeführt. Wenn Sie das Bonding wie oben beschrieben durchgeführt haben, müssen Sie keine weiteren Schritte unternehmen, der Installer weist Netzwerkkarten ohne eigene IP automatisch nicht zu. Sollten Sie jedoch eine andere Vorgehensweise gewählt haben und dadurch den „versklavten“ Interfaces eine (doppelte) IP zugewiesen haben, müssen Sie diese *unbedingt* auf „Do not use“ setzen, um bei Ausfall einer Verbindung nicht dem Cluster einen Störfall zu suggerieren.

Begründung: Der CRS arbeitet zwar akkumulierend, aber nicht redundant. D.h., jede hier angegebene Netzwerkkarte erhöht zwar die Bandbreite, führt aber bei ihrem Ausfall zu einer Split-Brain-Prevention im Cluster obwohl eigentlich noch eine zweite Verbindung bestünde. Um diesen Fall zu verhindern, haben wir die Redundanz zuvor schon auf Betriebssystemebene realisiert (NIC-Bonding, Punkt 2.2.2). Damit kommt zwar die erhöhte Bandbreite (des nun virtuellen Devices bond0) dem Cluster zugute, ein Ausfall jedoch wird durch den Bonding-Treiber abgedeckt. Nutzen Sie also nur *bond0* als private.

6. Geben Sie die (mindestens drei) unter Punkt 2.3.4 auf Seite 13 als Speicherort für die Voting Disks (Quoren) vorgesehenen Rawdevices an.
7. Geben Sie die beiden unter Punkt 2.3.4 auf Seite 13 als Speicherort für die Oracle Cluster Registry OCR vorgesehenen Rawdevices an.
8. Führen Sie das angeforderte *root.sh*-Skript auf allen Knoten nacheinander aus - es darf niemals auf beiden Knoten gleichzeitig laufen! Die Reihenfolge jedoch ist beliebig. Der Knoten, der das Skript zuerst ausführt, wird die OCR- und Voting Disks formatieren und seine lokalen Clusterdienste starten. Der zweite Knoten erkennt die bereits formatierten Volumes, überspringt deren Initialisierung und startet innerhalb von max. 10min auch bei sich die Clusterdienste.
9. Leider schlägt Cluvy und später auch der ab Release 2 ins CRS verlagerte VIPCA fehl, wenn Sie hier private IP-Adressen vergeben. Da etwas anders meist unsinnig wäre, muß dieser Fehler ignoriert und nachher manuell behoben werden.

Nun wäre die Installation des CRS eigentlich abgeschlossen, gäbe es da nicht diesen kleinen Störfall. Der Virtual IP Configuration Assistant (VIPCA) kann im Batchmodus keine privaten (oder auch „non-routeable“) IP-Adressen verarbeiten. Da die wenigsten unter Ihnen einen eigenen IP-Bereich besitzen werden, rufen Sie den VIPCA

nach Abschluß des (protestierenden) runInstallers manuell auf, und tragen in der sich öffnenden graphischen Oberfläche die IPs wie beschrieben nochmals ein. In diesem Modus akzeptiert der VIPCA die privaten Adressbereiche nämlich...

Nun bleibt noch die Wiederholung der durch den gescheiterten VIPCA abgebrochenen Aufgaben gemäß dem in der Abschlußfehlermeldung des runInstallers enthaltenen Skriptes, und schon sind Sie so gut wie fertig.

## 3.2 Abschluß

Es bleiben - wie immer - noch einige Restarbeiten für den Admin zu erledigen.

### 3.2.1 Überprüfung

Prüfen Sie, ob folgende Punkte erfüllt sind:

- Es wurde angelegt:
  - /etc/init.d/init.crs
  - /etc/init.d/init.crsd
  - /etc/init.d/init.cssd
  - /etc/init.d/init.evmd
- /etc/inittab enthält in ihren letzten drei Zeilen:

```
h1:35:respawn:/etc/init.d/init.evmd run >/dev/null 2>&1 </dev/null
h2:35:respawn:/etc/init.d/init.cssd fatal >/dev/null 2>&1 </dev/null
h3:35:respawn:/etc/init.d/init.crsd run >/dev/null 2>&1 </dev/null
```

Normalerweise trifft dies alles zu, und Sie sind stolzer Besitzer eines Clusters.

### 3.2.2 Bootfähigkeit herstellen

Jetzt muß dringend Sorge getragen werden daß das System wie folgt bootfähig gemacht wird, ansonsten besteht ernsthaft Gefahr einer unter Umständen sehr hartnäckigen Schleife. Gehen Sie folgendermaßen vor:

Fügen Sie als Kopf in die */etc/init.d/init.crs* die Informationen für den SuSE insserv wie in Algorithm 7 auf der nächsten Seite dargestellt ein. Nun entfernen Sie aus allen Runlevels manuell die symbolischen Links auf *init.crs* und führen dann im Verzeichnis */etc/init.d* den Befehl *insserv init.crs* durch.

**Algorithm 7** *init.crs*, Header

---

```

### BEGIN INIT INFO
# Provides: init.crs
# Required-Start: $local_fs $network $syslog raw
# Required-Stop:
# Default-Start: 3 5
# Default-Stop: 0 1 6
# Description: oracle cluster services un-flagging
### END INIT INFO

```

---

**Begründung** zur Startreihenfolge: Der CRS-Dienst benötigt zum Start a) Zugriff auf die Voting-Disks und b) einen bestehenden Cluster-Interconnect. Ist eine der beiden Bedingungen nicht erfüllt, spricht der OCSSD-host-kill an, und das System durchläuft einen Kaltstart. Stellen Sie daher sicher, daß vor Ausführung des Skriptes *init.crs* die Netzwerkadapter und die Rawdevices initialisiert wurden. Die Initialisierung der Rawdevices erfordert wiederum möglichen Zugriff auf die Storages über die Host-based adapters (HBA), bzw. auf die Devices die Sie gewählt haben. Dazu haben Sie auch schon Punkt 2.3.3 auf Seite 12 gelesen.

### 3.3 Starten und Stoppen

Die Clusterdienste zu starten und zu stoppen ist eine kleine Wissenschaft für sich. Gute Informationen dazu finden Sie bei Oracle Metalink in den Notes 263897.1, 265769.1 und 259301.1, die auch für Release 2 wenig an Aktualität verloren haben. Durch den Satz

„OCSSD (...) is armed with a node kill via the init script.“

seien Sie als der Administrator bei allem Folgenden zu größter Sorgfalt angehalten, um sich nicht unversehens in einer Bootschleife wiederzufinden.

In Kürze: Grundsätzlich werden die drei wirklichen Dienste *evmd.bin*, *ocssd.bin* und *crsd.bin* über ein respawn in den letzten drei Zeilen der Inittab (siehe 3.2.1 auf der vorherigen Seite) mit Umweg über die dort genannten Skripte gestartet. In diesen Skripten wird jedoch auf Flags geprüft, die von */etc/init.d/init.crs* umgesetzt werden. Die Einrichtung des automatischen Bootvorganges haben Sie ja schon gelesen, wollen Sie die Cluster-Dienste manuell starten und stoppen gilt jedoch zunächst folgendes:

„The (Oracle-)supported way to start the CRS is to boot the node.“

Unsupported oder im Notfall geht es aber auch anders. Versuchen Sie jedoch unter keinen Umständen, den laufenden Prozessen oder Skripten mit *kill* beizukommen, denn sie überwachen sich gegenseitig. Metalink spricht dazu:

„Thus (not necessarily optimally) killing either *init.cssd* or *ocssd* is fatal to the node.“

Gehen Sie also zum Stoppen des Clusterdienstes folgendermaßen vor:

```
/etc/init.d/init.crs stop # um die Flags umzusetzen  
init q # um init zu veranlassen, Ihre /etc/inittab neu einzulesen
```

Nun sollten nach ein, zwei Minuten in */var/log/messages* drei „*respawning too fast - disabled for 5 minutes*“-Fehlermeldungen von *init* auftauchen. Ihr Clusterknoten hat sich aus dem Verband herausgelöst.

Zum Start gehen Sie analog vor:

```
/etc/init.d/init.crs start # um die Flags umzusetzen  
init q
```

Letzteres diene dazu, durch das Einlesen der */etc/inittab* alle Prozesse neu zu respawnen die im Status *disabled* schlummerten. Ihr Cluster startet nun, und nach zwei bis drei Minuten müssen alle im Cluster registrierten Dienste laufen bzw. mindestens das Signal zum Start erhalten haben.

## 3.4 Überprüfung und Tests

Die CRS von Release 2 ist nun eine echte Clusterware, d.h. sie bietet erstmalig auch eine API bzw. Standardverfahren zur Absicherung von Drittanwendungen. Das Oracle Whitepaper „Using Oracle Clusterware to Protect 3rd Party Applications“ vom Mai 2005 (Autor: Philip Newlan) mag dazu eine gute Anleitung sein, hier ist der Rahmen dafür zu eng und es wäre uns sicherlich auch nur eine Wiederholung aus zweiter Hand möglich.

Jedoch sind zur Kontrolle vorangegangener Schritte ein paar Kleinigkeiten gut zu testen. Hier beweist sich (hoffentlich) der Wert der sorgfältigen Vorbereitung.

### 3.4.1 WICHTIG

Verwenden Sie für die Clustersteuerung niemals Binaries aus *\$ORACLE\_HOME/bin*! Einige (wie *srvctl*) sind zwar dort auch enthalten, aber laut OSS nur aus Kompatibilitätsgründen. Für alle *crs\_\**- und ähnliche Befehle (besonders *srvctl*!) verwenden Sie den Pfad, den Sie bei der Installation als Home für CRS angegeben haben, in unserem Beispiel wäre das */opt/oracle/product/10g/crs/bin*. Wir haben praktischerweise dafür eine Variable in der *oracle.sh* hinterlegt und im Folgenden auch verwendet: *\$ORA\_CRS\_HOME*. Setzen Sie Ihre *\$PATH*-Variable so, daß zuerst *\$ORA\_CRS\_HOME/bin* und dann erst *\$ORACLE\_HOME/bin* verwendet wird, wenn Sie für einen Befehl keinen Pfad angeben.

**Notabene:** Ein falsches Binary könnte Ihre OCR inkonsistent werden lassen und ein Recovery desselben notwendig machen.

### 3.4.2 VIPs

Um zu überprüfen, ob die Konfiguration der virtuellen IPs erfolgreich war, führen Sie

`$ORA_CRS_HOME/bin/crs_stat` aus. In der Ausgabe sollte

```
NAME=ora.node1.vip
TYPE=application
TARGET=ONLINE
STATE=ONLINE on node1
```

und

```
NAME=ora.node2.vip
TYPE=application
TARGET=ONLINE
STATE=ONLINE on node2
```

enthalten sein. Sie haben damit nachgewiesen, daß u.a. die virtuellen IPs als Dienst im Cluster registriert wurden. Die weiteren registrierten Dienste entnehmen Sie bitte ebenfalls dieser Liste. Dieses Kommando wird Ihnen bei einer späteren Fehlersuche der Clusterware Ihr steter Begleiter sein, da Sie vereinfacht ablesen können welcher Dienst gerade auf welchem Knoten läuft.

### 3.4.3 Service Relocation

Da noch kein Datenbanksystem und auch kein Listener läuft, können wir das Verschieben von Diensten/Ressourcen im Cluster sehr einfach mit der virtuellen IP testen. Dies ist notwendig, da ein späterer Fail-Over nichts als eine notfallmäßige Verschiebung darstellt. Sollte das schon „im Trockenen“ nicht funktionieren, besteht auch für den Notfall keine Hoffnung. Versuchen Sie daher, mit

`$ORA_CRS_HOME/bin/crs_relocate ora.node1.vip` die VIP vom Knoten 1 wegzuschieben. Auf welchem Knoten Sie den Befehl absetzen ist übrigens egal. Wenn er funktioniert, bekommen Sie folgende Rückmeldung:

```
node1:/ # crs_relocate ora.node1.vip
Attempting to stop 'ora.node1.vip' on member 'node1'
Stop of 'ora.node1.vip' on member 'node1' succeeded.
Attempting to start 'ora.node1.vip' on member 'node2'
Start of 'ora.node1.vip' on member 'node2' succeeded.
node1:/ #
```

Wenn Sie nun `crs_stat ora.node1.vip` ausführen, erhalten Sie erwartungsgemäß eine andere Meldung:

```
NAME=ora.node1.vip
TYPE=application
TARGET=ONLINE
STATE=ONLINE on node2
```

Wieder zurück verlagern Sie die VIP (versuchen Sie es ruhig vom anderen Rechner aus) erneut mit:

```
$ORA_CRS_HOME/bin/crs_relocate ora.node1.vip
```

Wenn Sie einen der Knoten herunterfahren, passiert genau dieser Vorgang. Die Dienste werden verlagert, die CRS gestoppt, und dann der Rechner stillgelegt. I.d.R. müssen Sie sich darum nicht kümmern, aber diesen Vorgang testen. Jetzt ist genau der richtige Zeitpunkt dafür.

Es hat funktioniert? Dann noch einmal, aber auf die harte Tour: Schalten Sie ein Node aus. Nach kurzer Zeit muß auf dem „überlebenden“ Knoten eine zusätzliche VIP auftauchen, und Ihre CRS teilen Ihnen das auf dem oben beschriebenen Weg mit. Wenn Sie das Gerät wieder einschalten, geben Sie der Anlage nach dem Hochfahren 10min Zeit, um sich wieder zu synchronisieren.

Alle Aktivitäten der Cluster Ready Services zeigt Ihnen übrigens

```
$ORA_CRS_HOME/log/<nodename>/crsd/crsd.log
```

#### 3.4.4 Gesamtstatus

Mit den Befehlen

```
$ORA_CRS_HOME/bin/srvctl status nodeapps -n node1 und srvctl status nodeapps -n node2 können Sie den aktuellen Status des Clusters prüfen. Dies sind die Services, deren Installation nach dem Abbruch des VIPCA laut dem Fehler-script des runInstallers manuell erfolgen sollte. Zum jetzigen Zeitpunkt dürfte die Ausgabe von Knoten 1 in etwa so aussehen, da der Listener erst mit dem RDBMS erstellt und registriert wird (Knoten 2 sinngemäß):
```

```
VIP wird auf Knoten ausgeführt: node1
GSD wird auf Knoten ausgeführt: node1
Listener wird nicht auf Knoten ausgeführt: node1
ONS-Daemon wird auf Knoten ausgeführt: node1
```

Weitere Kommandos und Ihre Wirkung stellen wir auch unter Punkt 8 auf Seite 36 vor. Hat ansonsten alles wie erwartet geklappt? Dann kommen wir nun zum Kern der Sache.

## 4 Installation Datenbank-System (RDBMS)

Sie benötigen einen geeigneten Installationssatz für Oracle Database 10g 64bit in der Version 10.2.0.1. Verwenden Sie dazu

*ship.db\_Disk1.linux86-64.cpio.gz* und *ship.db\_Disk2.linux86-64.cpio.gz* von Oracle TechNet und entpacken Sie sie als Benutzer oracle in Ihrem Installationsverzeichnis */opt/oracle/install* mit *gunzip* und *cpio* wie schon die CRS-Quellen unter Punkt 3.1 auf Seite 15. Es entstehen die Verzeichnisse *./Disk1* und *./Disk2*. Damit müssen Sie während der Installation keinen Medienwechsel vornehmen, diese Struktur wird automatisch durchsucht.

1. Gehen Sie als Benutzer oracle in das eben erzeugte Verzeichnis *./Disk1* und starten Sie das Script *./runInstaller* mit gültiger \$DISPLAY-Variable Ihrer Shell.
2. Wichtig: Vergeben Sie für OUIhome2 ein anderes Verzeichnis als das zuvor für CRS vergebene. Wählen Sie beispielsweise */opt/oracle/product/10g/db*
3. Wählen Sie die Komponenten, die Sie benötigen und für die Sie lizenziert sind zur Installation aus.
4. Geben Sie an, keine Datenbank erzeugen zu wollen.
5. Nach der Installation und dem Linking der Komponenten fordert der runInstaller die Ausführung des *\$ORACLE\_HOME/root.sh* auf beiden Knoten an. Dieses Skript hatte bis Release 2 den VIPCA gestartet, diese Aktivität wurde nun jedoch in die Clusterware verlegt und entfällt demnach an der Stelle.
6. Starten Sie auch auf dem 2. Knoten als Benutzer root *\$ORACLE\_HOME/root.sh*. Wichtig: *root.sh* darf nicht auf beiden Knoten gleichzeitig aktiv sein, die beiden Skripte müssen sich nacheinander ergänzen können.
7. Damit ist die Installation des RDBMS abgeschlossen.

## 5 Instanzen

Die Namesvergabe für unser Beispiel entnehmen Sie bitte Tabelle 3 auf der nächsten Seite, denn immer mit abstrakten Begriffen zu hantieren ist der Lesbarkeit sicher nicht zuträglich.

Tabelle 3: Instanzen - Namensvergabe

Knoten	Instanz-Name	Funktion
node1	+ASM1	OU, Zugriff auf Storage
node1	TESTR1	Nutzinstanz des Clusters
node2	+ASM2	OU, Zugriff auf Storage
node2	TESTR2	Nutzinstanz des Clusters

Als globalen Datenbanknamen haben wir „*TESTRAC*“ gewählt.

Die Bezeichnung unserer ASM-Diskgroup lautet „*+TESTRACVOL*“.

## 5.1 Theorie

Die hier genannten Unterpunkte sollen Ihnen einen kleinen Überblick bezüglich der Zusammenhänge im Cluster geben. Erfahrene 10g-Admins mögen 5.1 bitte großmütig überlesen und bei 5.2 auf Seite 26 wieder einsteigen.

### 5.1.1 ASM- und Nutzinstanz

Grundsätzlich sprechen wir bei einem RAC mit ASM über mindestens doppelt so viele Instanzen wie Knoten im Cluster vorhanden sind. Denn: jeder Knoten betreibt als organisatorischen Layer zwischen Hardwarezugriff und Nutzdatenbank eine eigene ASM-Instanz. Diese ermöglicht der „Nutz“-Instanz Zugriff auf das unterliegende Shared-Storage-System. ASM hält darüber hinaus in unserem Setup beim Schreiben die beiden Spiegel konsistent und holt beim Lesezugriff Daten aus beiden Storages parallel.

**Merke:** „die“ Clusterinstanz existiert nicht, das Cluster besteht „nur“ aus dem Zusammenwirken der Nutzinstanzen der einzelnen Knoten.

Aus dem engen Verhältnis zwischen ASM- und Nutzinstanz auf den Knoten folgt messerscharf, daß zur Laufzeit stets eine ASM-Instanz vorhanden sein muß. Wird z.B. im Betrieb +ASM1 beendet, erhält die darauf aufbauende Nutzinstanz TESTR1 ein unfreundliches *shutdown abort*.

Das Erstellen der Instanzen mit dem Database Configuration Assistant *dbca* trägt all dem Rechnung, und erzeugt zuerst die benötigten ASM-Instanzen. Sollten Sie die Erzeugung der Nutzinstanzen später wiederholen müssen, werden Sie den Wegfall einiger ASM-spezifischer Schritte bemerken.

### 5.1.2 Diskgroup/Failgroup

Die Zusammenfassung von Massenspeichern zu einem nutzbaren Verband wird im Kontext ASM eine *Diskgroup* genannt. Für unser Beispiel wählen wir nur eine einzige solche, ihr Name sei *+TESTRACVOL*. In dieser Diskgroup liegen neben unseren Nutzdaten noch weitere betriebskritische Informationen. Sie setzt sich aus einzelnen *Failgroups* zusammen. Eine *Failgroup* ist eine Organisationseinheit, mit der erwartungsgemäß gleichzeitig von einem Ausfall betroffene Massenspeicher zusammengefasst werden. Damit erhält ASM schon im Vorfeld eine Information, wo Daten abhängig vom Redundanzmodus mirrored (zwischen den Failgroups) und wo striped (innerhalb einer Failgroup) gespeichert sein sollen. Per default wird jedes hinzugefügte Volume einer eigenen Failgroup zugeordnet.

### 5.1.3 spfile

Das *spfile* ist die binäre Startinformation der Instanz. Sein Pfad für eine „Nutzinstanz“ lautet z.B.

„*+TESTRACVOL/TESTRAC/spfileTESTR.ora*“. Wie sein Inhalt verändert werden kann, lesen Sie später noch unter Punkt 6.2.1 auf Seite 34. Das *spfile* wird beim Start für alle Nutzinstanzen des Clusters angewandt, ggf. zu Laufzeit geändert und muß deshalb auf Shared Storage liegen.

### 5.1.4 undo-Tablespaces

Dies trifft aus ähnlichen Gründen auch für die undo-tablespaces der Instanzen zu.

**Beispiel:** node2 befindet sich beim Schreibvorgang und wird noch vor einem *commit* „abgewürgt“. node1 muß nun zur Wahrung der Datenintegrität einen *rollback* ausführen. Läge *undotbs2* auf lokaler Disk von node2, wäre dies unmöglich.

Ergo: In *+TESTRACVOL* drängt sich für jede Instanz ein eigener Undo-Tablespace.

### 5.1.5 redo-Logs

Ein Szenario, in dem ein *database recovery* knotenübergreifend notwendig wird läßt sich ebenso leicht ausmalen, so daß die redo-Logs ebenfalls mindestens einmal im Shared-Storage-Bereich abgelegt werden müssen.

### 5.1.6 Cluster

Die eigentliche Clusterfunktionalität mit Loadbalancing und Ausfallschutz des RAC ergibt sich erst in Zusammenarbeit mit dem Client über Listener und/oder Dispatcher. Zwar nutzt auch z.B. das externe Oracle-Tool RMAN Clusterfunktionen, die Mitwirkung des eigentlichen RAC beschränkt sich jedoch im Wesentlichen darauf,

synchron auf jedem der Knoten gleiche Informationen bereitzustellen. Er realisiert das durch CacheFusion über den Cluster-Interconnect und den schon mehrfach erwähnten Shared-Storage-Zugriff. Dazu kommen dann noch aufeinander abgestimmte Listener, Dispatcher und ähnliches. Zu Details möchte ich Sie aber auf Punkt 6 vertragen.

Sind Sie bereit? Dann nichts wie los.

## 5.2 Instanzerzeugung mit dbca

Wir wollen Ihnen hier erläutern, wie Sie mit dem Database Configuration Assistant *dbca* eine Clusterdatenbank ins Leben rufen. Der Vorgang basiert auf den zuvor genannten Grundlagen und ist abhängig von der sicheren Funktion der unterliegenden Clusterinfrastruktur CRS.

Gehen Sie zur Erzeugung der Datenbank wie folgt vor:

1. Öffnen Sie eine Shell als Benutzer oracle und gültiger \$DISPLAY-Variable und starten Sie  
*/opt/oracle/product/10g/bin/dbca*.
2. Wenn Ihre CRS richtig funktionieren und gestartet sind, erkennt dbca dies und fragt nach dem Datenbanktyp. Wählen Sie hier „Oracle Real Application Clusters-Datenbank“ und im nächsten Schritt „Datenbank erstellen“. Wenn Sie nur eine ASM-Instanz erzeugen möchten, bietet der *dbca* seit Release 2 hierzu an dieser Stelle eine Option an. „Datenbank erstellen“ schließt jedoch diesen Schritt auch künftig mit ein.
3. Nun werden Sie gefragt, auf welchen Knoten Ihres Clusters die Erzeugung erfolgen soll. Geben Sie beide Knoten (node1, node2) an.
4. Wählen Sie nun entweder ein vorkonfiguriertes Template, oder „Benutzerdefinierte Datenbank“. Wir wählten hier letzteres, da einige Feineinstellungen notwendig sind.
5. Tragen Sie als globalen Datenbanknamen z.B. „*TESTRAC*“ ein und wählen als SID-Präfix „*TESTR*“.
6. Wählen Sie die in Ihrer Umgebung angebrachten Parameter bzgl. Enterprise-Manager / Grid-Control, Benachrichtigung und Backup.
7. Vergeben Sie Ihre Benutzer-Kennwörter.
8. Wählen Sie als Speicherungsverfahren „ASM“
9. Vergeben Sie ein Passwort für den Benutzer sys der ASM-Instanz. Geben Sie außerdem unter „ASM-Parameter“ *asm\_power\_limit=5* an. Damit legen Sie einen Default-Wert für die Beschleunigung eines eventuellen Rebalancings nach Verlust und Wiederherstellung einer Storage an. Laut Oracle Support Services

bringt eine Einstellung auf Werte größer als „5“ keine Skalierung der Leistung mehr, die möglichen Probleme wachsen dagegen exponential. Unserer Erfahrung nach ist das Recovern einer verlorenen Platte in ASM mit rebalance power 11 quasi unmöglich.

10. *dbca* fragt Sie nach möglichen Plattengruppen. Klicken Sie auf „Neue erstellen“.
11. Vergeben Sie den Namen „*TESTRACVOL*“ (hier ohne „+“ am Anfang) und wählen Sie Redundanz „normal“. Damit spiegelt ASM symmetrisch auf beide Member. Im unteren Teil sollten automatisch Ihre geeigneten Rawdevices als Kandidaten erscheinen. Ist dies nicht der Fall, sollten Sie sicherstellen daß alle Rawdevices gemapped wurden und ggf. auf „alle anzeigen“ zurückgreifen. Nutzt auch das nichts, können Sie noch den Disk-Discovery-Pfad so abändern, daß er auf das Verzeichnis Ihrer Rawdevices zeigt. Seien Sie vorsichtig, hier nicht versehentlich die OCR- oder Voting Disks anzugeben. Ihre Clusterware würde es übelnehmen.
12. Nun erzeugt *dbca* die ASM-Instanz, und Sie werden nach dem Speicherort Ihrer Datenbank gefragt. Hier erscheint nun die soeben erzeugte Diskgroup „*TESTRACVOL*“. Markieren Sie sie.
13. Stellen Sie im nächsten Fenster die Speicherziele ein: Wählen Sie Oracle Managed Files und tragen Sie für den Datenbankbereich und für ein Redo-Log- und Kontrolldateienziel „+*TESTRACVOL*“ ein. Möchten Sie die Redologs und Kontrolldateien noch auf einem zweiten Platz spiegeln, ist nun die Zeit dafür diesen Ort anzugeben.
14. Tragen Sie für den Flash-Recovery-Bereich ebenfalls „+*TESTRACVOL*“ ein, und suchen Sie sich ein gutes Ziel für Ihre Archivelogs, falls Sie den Archivelog-Modus benötigen.
15. Nun fragt *dbca* die Initialisierungsparameter ab. Vieles davon sind Tuningparameter und damit Geschmacks- bzw. Bedarfssache. Wir stellen Ihnen hier unsere Werte nur als Anhaltspunkte dar.
  - (a) Speicher - Speicherverteilung haben wir für den Einstieg auf Standard - 75% gesetzt haben und sind damit gut gefahren. Unsere Systeme haben je 8GB RAM. Vorsicht - eine 32bit-Maschine (Off-topic) kann hier nur 2GB zuweisen und wird ansonsten bei der Erzeugung der Instanz einen ORA-27123 präsentieren. Nur der Vollständigkeit halber möchten wir erwähnen, daß hier natürlich nur zur Verfügung steht, was durch den Kernelparameter SHMMAX ermöglicht wurde!
  - (b) Skalierung - für uns war Blockgröße 8kB und eine Prozessanzahl von 600 die richtige Wahl.
  - (c) Zeichensatz - WE8MSWIN1252, Länderspezifischer Zeichensatz AL16UTF16, Sprache GERMAN, Datumsformat GERMANY.

- (d) Verbindungsmodus - dedicated. Hier werden wir später unter 6 auf der nächsten Seite noch manuell nacharbeiten, diese Einstellung ist nur von wenig Bedeutung für das Gesamtergebnis.
16. Erstellen Sie nun das für Ihre Zwecke benötigte Datenbankprofil. Der dbca hat hier einige Eigenwilligkeiten oder besser gesagt Schattenseiten: Gehen Sie nie davon aus, daß Werte geändert werden, wenn Sie das Eingabefeld nicht mit <tab> verlassen haben, und löschen Sie nie *alle* RedoLog-Gruppen, Sie werden nämlich dann keine mehr hinzufügen können.
  17. Tablespaces - Überschreiten Sie ohne Not die Grenze von 4GB pro Datenfile nicht. Oracle Support Services bestreitet zwar die Existenz einer 4GB-Grenze, wir hatten jedoch gelegentlich Probleme damit, deren Ursache nicht geklärt werden konnte.
  18. RedoLogs - erzeugen Sie für jeden Knoten mindestens sechs Gruppen von RedoLogs, jeder Knoten stellt einen eigenen *Thread* dar. Vorsicht, die Default-Nummerierung ist etwas eigenwillig und kann sich durch Vorwärts- und Rückwärtsblättern im dbca schon einmal ändern. Als Dateigröße hat sich hier für uns 204800K bewährt. Allerdings muß diese Entscheidung jeder Admin selbst treffen - 200MB sind für einen einzelnen Logswitch viel Holz, aber auf einem schwer schreibbelasteten System kann damit die Zeit zwischen den Logswitches ein hochgehalten werden und damit die Gesamtperformance etwas steigen.
  19. Speichern Sie Ihre Konfiguration nun als Template ab und starten Sie das Erzeugen der Datenbank. Dies kann abhängig von der Größe Ihrer Dateien ein paar Viertelstunden dauern und ist die Krönung Ihrer (und unserer) Anstrengungen. Hier rächen sich aber alle Fehler, die während der Vorbereitungszeit gemacht wurden. Ein Ausschnitt aus unserer „Trophäensammlung“ an dieser Stelle:
    - (a) ORA-27123 Unable to attach shared memory segment (Speicheradressierung möglich?)
    - (b) ORA-01034 Oracle not available. (Grundsätzlich Start der Instanz fehlgeschlagen - warum auch immer)
    - (c) ORA-00200 Kontrolldatei konnte nicht erstellt werden (ASM-Problem?)
    - (d) ORA-15001 Plattengruppe nicht vorhanden oder nicht gemountet (ASM-Problem?)
    - (e) ORA-15055 Anmeldung bei ASM-Instance nicht möglich
    - (f) ORA-01031 Insufficient privileges (ist oracle Mitglied in Gruppe dba?)
    - (g) und so fort. :-)
  20. Nach erfolgreichem Erstellen der Instanz meldet der dbca Vollzug, bietet eine Möglichkeit zur Benutzerverwaltung und beendet sich nach Drücken von „Ok“.

Wenn Sie so weit gekommen sind, kann nichts mehr schiefgehen. Nur Arbeit wartet noch.

## 5.3 Überprüfung

Melden Sie sich auf beiden Knoten als Benutzer oracle an, und führen Sie *sqlplus '/ as sysdba'* aus. Sie sollten wie folgt belohnt werden:

```
oracle@node1:~> sqlplus / as sysdba
SQL*Plus: Release 10.2.0.1.0 - Production on Di Jan 10
Copyright (c) 1982, 2005, Oracle. All rights reserved.
Verbunden mit:
Oracle Database 10g Enterprise Edition
Release 10.2.0.1.0 - 64bit Production
With the Partitioning, Real Application Clusters
and Data Mining options
SQL>
```

Die Abfrage *select instance\_name,status from v\$instance;* liefert Ihnen nun abhängig vom Knoten TESTR1 oder TESTR2 und OPEN zurück.

Nun können Sie erstmals einen Import eines Schemas testen, ein RMAN-Backup zurückspielen oder einfach nur eine Tabelle anlegen. Testen Sie, ob eingespielte Daten auf beiden Knoten verfügbar werden - Sie werden begeistert sein. Jetzt fehlt „nur“ noch das korrekte Zusammenspiel aller Komponenten.

Auf zum Endspurt!

## 6 Clusterfunktionalität

Wie schon zuvor erwähnt, wird das Cluster erst hochverfügbar und nutzt seine Ressourcen parallel, wenn der Clientzugriff korrekt konfiguriert wurde. Wir haben für unser Setup

- TAF (Transparent Application Failover) mit
- dediziertem und shared Zugriff,
- Clientseitigem Failover und
- Serverseitigem Loadbalancing

gewählt und werden nur diese Kombination hier vorstellen.

### 6.1 Theorie

Zunächst möchten wir Ihnen eine kleine Einführung in die Zusammenhänge der Zugriffsmethodik geben.

### 6.1.1 Listener

Der Dreh- und Angelpunkt des Zugriffs sind die Listener auf den Knoten. Sie bieten einen Netzwerksocket (Default-Port ist 1521) für Datenbankverbindungen auf den virtuellen IP-Adressen (warum, bitte unter Punkt 6.1.3 auf Seite 32 nachschlagen). Diese nutzen das TNS (Transparent Network Substrate)-Protokoll. Der Listener wird mit dem Befehl *lsnrctl* gestartet und gestoppt. Beispiel:

```
lsnrctl start listener_node1
```

Listener, die als Service (aka Ressource) im Cluster registriert sind, sollten Sie idealerweise mit

```
srvctl [start|stop] listener -n <Knotenname>
```

starten und stoppen, da ansonsten die CRS nichts von einem Stop den Listeners wissen und ihn umgehend wieder starten.

Die Listener werden auf jedem Knoten einzeln in der *\$ORACLE\_HOME/network/admin/listener.ora* konfiguriert. Später zeigt Algorithm 9 auf Seite 33 dazu ein Beispiel.

### 6.1.2 Clientzugriff

Sie greifen i.d.R. mit Hilfe von *sqlplus* oder mit Hilfe der Oracle-Bibliotheken aus der Oracle10g-Clientinstallation über das TNS-Protokoll auf den Listener des Servers zu. Im einfachsten Fall lautet der Befehl dazu etwa:

```
sqlplus 'system/passwort@servicename'
```

Die Konfiguration des Clients, z.B. des Servicenamens, erfolgt in der *\$ORACLE\_HOME/network/admin/tnsnames.ora* des Clients.

In der ersten Zeile von Algorithm 8 auf der nächsten Seite, der ein Beispiel für die Parametrierung Ihres Zugriffs auf die Listener zeigt, sehen Sie zunächst einen Servicenamen unter dem Sie Ihre Konfiguration nutzen können. Dieser Name ist vom Server unabhängig und Sie können ihn den Bedürfnissen auf dem Client entsprechend beliebig anpassen.

**Algorithm 8** tnsnames.ora für dedizierten Zugriff

---

```

TESTR1.EXAMPLE.COM =
  (DESCRIPTION =
    (LOAD_BALANCE=off)
    (FAILOVER=on)
      (ADDRESS = (PROTOCOL=TCP) \
        (HOST = node1-vip.EXAMPLE.COM) (PORT=1521))
      (ADDRESS = (PROTOCOL=TCP) \
        (HOST = NODE2-vip.EXAMPLE.COM) (PORT=1521))
    (CONNECT_DATA =
      (SERVER = DEDICATED)
      (SERVICE_NAME = TESTRAC)
      (FAILOVER_MODE =
        (TYPE = SELECT)
        (METHOD = BASIC)
        (RETRIES = 120)
        (DELAY = 2)
      )
    )
  )

```

---

**LOAD\_BALANCE** für das clientseitige Loadbalancing (das ja nur auf dem Round-Robin-Verfahren beruht) böte nur statistisch eine Lastverteilung und ist deshalb hier ausgeschaltet.

**FAILOVER** beruht auf der Fähigkeit der Listener der beiden Knoten, die aktuell herrschenden Lastverhältnisse auf allen Nodes zu kennen und davon abhängig die Verbindungsanfrage zu akzeptieren oder auch abzulehnen. Im letzteren Fall versucht der Client ohne jeden Verzug den nächsten hier konfigurierten Host zu erreichen. Dies geschieht auch im eigentlichen Fail-Over-Fall so, wenn einmal eines der Ziele schon beim Verbindungsaufbau nicht erreichbar sein sollte.

**CONNECT\_DATA** spezifiziert die näheren Details der Verbindung. Hier wird zwischen DEDICATED und SHARED unterschieden.

Dedizierte Verbindung bedeutet, daß jeder Zugriff eine eigene TCP-Verbindung aufbaut und damit auf dem Server seinen eigenen Serverprozess forkt.

Bei „shared“ dagegen nutzen mehrere Zugriffe auf die Datenbank einen gemeinsame Serverprozess: die ganze Angelegenheit läuft nach dem Listener über den zuvor konfigurierten Dispatcher.

**FAILOVER\_MODE** legt Parameter für den Fall einer schon aufgebauten, aber später abreißen Verbindung fest.

Für TYPE gelten SELECT (Verbindung und Lesezugriffe laufen auf dem nächsten Knoten weiter, Schreibzugriffe brechen ohne *commit* ab), SESSION (nur die Verbindung wird weitergereicht, Statements unterbrechen) und NONE (kein Fail-Over zu Laufzeit).

Als METHOD können Sie zwischen BASIC (Neuaufbau der Verbindung zum Ausweichknoten bei Abriß der ersten Verbindung) und PRECONNECT (vorsichtshalber wird schon „zu Lebzeiten“ eine Ausweichverbindung gepflegt) wählen.

RETRIES ist die Anzahl der Versuche zum Neuaufbau einer abgerissenen Verbindung, bevor auf die Zweite umgestellt wird, und

DELAY ist die Wartezeit zwischen diesen Versuchen. Bei in unserem Beispiel ergibt sich demnach eine Wartezeit von  $120 * 2s = 4min$ .

### 6.1.3 VIPs und TAF

Der durch RETRIES und DELAY eingestellte Timeout greift immer dann, wenn eine Verbindung keine Antwort mehr liefert, z.B. bei Ausfall eines Knotens. Bedenken Sie, daß alle TNS-Aktivitäten auf Layer 5 des ISO/OSI-Modells stattfinden und unterliegende TCP-Verbindungen deshalb von TNS-Einstellungen unberührt bleiben. Beide Layer verursachen jedoch gleichzeitig ähnliche Probleme. Lassen Sie uns deshalb zur Klärung kurz ausschweifen und einen möglichen Fehlerfall, den Knotenausfall im Cluster, näher erläutern.

Der Client hat eine bestehende TNS- (und damit auch TCP-)Verbindung zu node1. node1 fällt aus. Was passiert?

1. Die TCP-Verbindung würde laut Spezifikation der RFC793 10 Timeout-Minuten lang immer wieder versuchen, ob sich am Knoten nicht doch wieder etwas tut. Das ist jedoch bei einem katastrophalen Ausfall unsinnig und führt nur zu blockierten Applikationen. Deshalb übernimmt node2 so schnell als möglich die virtuelle IP des node1. Wir haben hier Zeiten von rund 40 Sekunden realisiert.
2. Nun kennt jedoch der TCP-Stack des node2 die aktuellen Verbindungsdaten des node1 nicht, empfängt aber trotzdem ständig die Anfragen des Clients an diesen. Er wird deshalb im Stadium nach der Übernahme der VIP von node1 jedes TCP-Paket an die VIP von node1 mit einem TCP-Flag „RST“ beantworten.
3. Dies führt auf Clientseite wiederum zum Abbruch der Verbindung. Aber ist das nun gut oder schlecht? Es ist gut, denn dadurch geht der Client in der in den Algorithms 8 und 11 gezeigten Hostliste („ADDRESS“-Zeilen) um eine Zeile weiter und baut eine TCP-Verbindung (Layer 4) mit dem nächsten Knoten auf.
4. Bei dieser Umschaltung, die bei unseren Systemen maximal 90 Sekunden dauerte, wird die TNS-Verbindung auf Layer 5 nicht gestört - sie hat ja 4 Minuten Timeout bei einem Prüfintervall von 2 Sekunden. Auf der Clusterseite werden die dazu nötigen TNS-Informationen (Layer 5) durch die Listener abgepuffert, die stets die hierfür notwendigen Informationen untereinander austauschen.

Ein eventuell noch laufendes SELECT-Statement wird bei entsprechender Konfiguration dank CacheFusion den Wechsel der TCP-Verbindung von seiner TNS-Session gut geschützt überstehen und nach rund 90 Sekunden einfach weiterlaufen.

Nun aber genug der Theorie, geben wir Ihrem Cluster in der Praxis den letzten Schliff.

## 6.2 Einrichten

### 6.2.1 Server

Am Anfang war der Listener - wenn er nicht läuft, läuft nichts. Ersetzen Sie deshalb die

`$ORACLE_HOME/network/admin/listener.ora` von node1 durch den Inhalt von Algorithm 9 und tun Sie gleiches für node2 unter sinngemäßer Abänderung der Parameter.

---

#### Algorithm 9 listener.ora node1

---

```

LISTENER_node1 =
  (DESCRIPTION_LIST =
    (DESCRIPTION =
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = TCP) \
          (HOST = node1-vip.EXAMPLE.COM) (PORT = 1521))
      )
      (ADDRESS_LIST =
        (ADDRESS = (PROTOCOL = IPC) (KEY = EXTPROC))
      )
    )
  )
SID_LIST_LISTENER_node1 =
  (SID_LIST =
    (SID_DESC =
      (SID_NAME = +ASM1)
    )
  )

```

**Anmerkung:** Der zweite Teil (SID\_LIST) dient nur dem Zugriff des Enterprise-Managers auf die ASM-Instanz des Knotens, er ist für die Clusterfunktionen ohne Belang.

---

Starten Sie im Anschluß den Listener auf Ihrer Shell als Benutzer oracle.

```
lsnrctl start listener_node1
```

Nun muß der Instanz mitgeteilt werden, bei welchem Listener sie sich registrieren muß. Neben dem *alter*-Kommando unter *sqlplus*

```
alter system set <parameter>='<wert>' scope='both';
```

können Sie das auch über eine indirekte Veränderung des *spfiles* erledigen. Da dies bei einem *spfile* auf shared storage diffizil und auch in anderen Situationen dienlich sein kann, sei hier kurz darauf eingegangen.

Stoppen Sie die Nutzinstanzen auf beiden Knoten unter *sqlplus '/ as sysdba'* und exportieren Sie in *sqlplus* das *spfile* als *pfile*.

**Anmerkung:** Geben Sie hier unbedingt einen Pfad für das *pfile* an, ansonsten wird per default Ihr *initfile* *\$ORACLE\_HOME/dbs/initTESTR1.ora* für die Instanz überschrieben, und Sie müssen sich den dort notwendigen Link auf das *spfile* (SPFILE='+TESTRACVOL/TESTRAC/spfileTESTR.ora') wieder per Hand zurechtmachen.

```
shutdown immediate;
create pfile='/opt/oracle/pfile' from spfile;
```

Nun verändern Sie das *pfile*, das als ASCII-Datei vorliegt, wie in Algorithm 10 dargestellt. Löschen Sie die gesamten Listener- und Dispatcher-Einträge bis auf die gezeigten Zeilen. Die Zusammenfassung zur Gruppe LISTENERS\_TESTRAC wurde ja auch schon in der *listener.ora* in dieser Form dem Listener angegeben.

---

#### Algorithm 10 pfile TESTRAC

---

```
*.remote_listener='LISTENERS_TESTRAC'
TESTR1.local_listener='LISTENER_node1'
TESTR2.local_listener='LISTENER_node2'
*.dispatchers='(PROTOCOL=TCP) (DISPATCHERS=4) (SERVICE=TESTRXDB)'
```

---

Jetzt übernehmen Sie das *pfile* wieder als *spfile*

```
create spfile='+TESTRACVOL/testrac/spfileTESTR.ora' \
from pfile='/opt/oracle/pfile';
```

Starten Sie nun die Instanzen auf beiden Knoten und registrieren Sie sie manuell bei den Listeners.

```
startup;
alter system register;
```

War dies erfolgreich, liefert *lsnrctl status listener\_node1* etwa diese Rückgabe, wobei natürlich auf node2 die Ausgabe ähnlich sein sollte:

```
Services Übersicht...
Dienst "+ASM1" hat 1 Instance(s).
  Instance "+ASM1", Status UNKNOWN, hat 1 Handler für diesen Dienst...
Dienst "TESTRAC" hat 2 Instance(s).
  Instance "TESTR1", Status READY, hat 2 Handler für diesen Dienst...
  Instance "TESTR2", Status READY, hat 1 Handler für diesen Dienst...
Dienst "TESTRXDB" hat 2 Instance(s).
  Instance "TESTR1", Status READY, hat 4 Handler für diesen Dienst...
  Instance "TESTR2", Status READY, hat 4 Handler für diesen Dienst...
Der Befehl wurde erfolgreich ausgeführt.
```

Bon? Bon! Serverseitig ist nun alles im Lot.

### 6.2.2 Client

Fügen Sie Algorithm 8 auf Seite 31 für eine dedizierte Verbindung in Ihre *\$ORACLE\_HOME/network/admin/tnsnames.ora* auf dem Client ein.

In der Regel wird man noch eine zweite Verbindung, z.B. TESTR2.EXAMPLE.COM mit der umgekehrten Hostreihenfolge hinzufügen, oder beide Verbindungen nicht dediziert sondern shared einrichten. Für die shared-Variante siehe Algorithm 11 auf der nächsten Seite und die Erklärung zu CONNECT\_DATA auf Seite 31.

**Algorithm 11** tnsnames.ora für shared Zugriff

---

```

TESTR1_SHARED.EXAMPLE.COM =
  (DESCRIPTION =
    (LOAD_BALANCE=off)
    (FAILOVER=on)
      (ADDRESS = (PROTOCOL=TCP) \
        (HOST = node1-vip.EXAMPLE.COM) (PORT=1521))
      (ADDRESS = (PROTOCOL=TCP) \
        (HOST = node2-vip.EXAMPLE.COM) (PORT=1521))
    (CONNECT_DATA =
      (SERVER = SHARED)
      (SERVICE_NAME = TESTRAC)
      (FAILOVER_MODE =
        (TYPE = SELECT)
        (METHOD = BASIC)
        (RETRIES = 120)
        (DELAY = 2)
      )
    )
  )

```

---

## 7 Fail-Over, Recovery

## 8 Nützliche Kleinigkeiten

Wir haben ein paar Kleinigkeiten für Sie zusammengetragen, die in mancher Situation hilfreich sind. Eine wirklich umfassende Befehlsreferenz würde jedoch den Rahmen dieses Dokuments deutlich sprengen.

**Vorsicht:** Beachten Sie in *jedem Fall die Hinweise* unter Punkt 3.4.1 auf Seite 20! Ein Schreibvorgang mit dem falschen Binary kann Ihre OCR auf Nimmerwiedersehen schädigen.

### 8.1 srvctl

Für das Starten und Stoppen verschiedener Clusterfunktionen im Betrieb ist das Werkzeug *srvctl* unverzichtbar. Das Tool hat den Vorteil, die Starts und Stops über die CRS-Schnittstellen durchzuführen. Damit testen sie a) diese im Notfall wichtigen Kanäle, und b) wird der Clusterdienst mit seinen Watchdog-Funktionen benachrichtigt, daß ein Wartungseingriff vorliegt. Im Gegensatz zu den unter Punkt 8.2 auf Seite 38 vorgestellten *crs\_\**-Tools sind die *srvctl*-Eingriffe durch OSS supported.

Das Binary liegt auf *\$ORA\_CRS\_HOME/bin/srvctl*.

### 8.1.1 Instanz

Die Statusabfrage

```
srvctl status database -d TESTRAC
```

liefert im Idealfall

```
Instance TEST1 wird auf Knoten node1 ausgeführt  
Instance TEST2 wird auf Knoten node2 ausgeführt
```

wenn nicht,

```
srvctl start instance -d TESTRAC -i TEST1[,TEST2]
```

startet Instanz TEST1 [und ggf. TEST2]. Die Datenbank (-d) ist stets mit anzugeben. Analog funktioniert:

```
srvctl stop instance -d TESTRAC -i TEST1[,TEST2]
```

Für beide Kommandos bitte etwas Geduld mitbringen, sie können gar nicht schneller sein als *SQL> startup*; oder *SQL> shutdown immediate*;!

### 8.1.2 ASM

Die Statusabfrage

```
srvctl status asm -n node1
```

liefert idealerweise

```
ASM Instance +ASM1 wird auf Knoten node1 ausgeführt.
```

zurück. Analog natürlich für alle weiteren Knoten. Ist dies einmal nicht der Fall oder ist dies unerwünscht, helfen

```
srvctl start asm -n node1
```

oder

```
srvctl stop asm -n node1
```

weiter.

### 8.1.3 Ressourcen und Hilfsdienste

Zur Abfrage der Ressourcen Virtuelle IP (VIP), GSD, Listener und ONS auf Knoten 1 dient

```
srvctl status nodeapps -n node1
```

und liefert ggf. folgende Antwort:

```
VIP wird auf Knoten ausgeführt: node1
GSD wird auf Knoten ausgeführt: node1
Listener wird auf Knoten ausgeführt: node1
ONS-Daemon wird auf Knoten ausgeführt: node1
```

## 8.2 Ressourcenverwaltung mit *crs\_\**

### 8.2.1 Status anzeigen mit *crs\_stat*

*crs\_stat* liefert zu allen Ressourcen des Clusters Rohinformationen. Dabei handelt es sich beispielsweise um alle Instanzen, Listener, Dispatcher, Virtuelle IPs etc.

Sie finden das Binary in Ihrem `$ORA_CRS_HOME/bin`. Eine Eingabe des „nackten“ Befehls liefert eine Liste der zur Zeit im Cluster registrierten Ressourcen:

```
NAME=ora.TESTRAC.TEST1.inst
TYPE=application
TARGET=ONLINE
STATE=ONLINE on node1
NAME=ora.TESTRAC.TEST2.inst
TYPE=application
TARGET=ONLINE
STATE=OFFLINE
...
```

Mit *crs\_stat <service>* erhalten Sie eine Einschränkung der Liste auf die angegebene Ressource, z.B.

```
crs_stat ora.node1.LISTENER_NODE1.lsnr
```

liefert

```
NAME=ora.node1.LISTENER_NODE1.lsnr
TYPE=application
TARGET=ONLINE
STATE=ONLINE on node1
```

### 8.2.2 Manipulieren mit `crs_*`

Etwas Vorsicht ist in diesem Bereich dringend anzuraten: Oft sind Maßnahmen hier „heavily unsupported“! Wir empfehlen, für die Oracle-Ressourcen auf Tools wie `crs_relocate` verzichten wenn nicht WIRKLICH dringende Gründe dafür sprechen.

- `crs_start <ressource>` startet die angegebene Ressource
- `crs_stop <ressource>` stoppt sie
- `crs_register <ressource>` fügt sie dem Cluster Repository (OCR) hinzu
- `crs_unregister <ressource>` entfernt sie aus dem OCR
- `crs_relocate <ressource> -c <knoten>` verschiebt die Ressource im Cluster (VORSICHT!)

Jeder der Befehle liefert eine kurze Übersicht über seine Syntax, wenn er ohne Parameter aufgerufen wird.

## 9 Editorial

Dieses Dokument entstand im Rahmen eines Projektes zur Performancesteigerung und Absicherung der Intranet-Lösung „AWIS“ im Hause A.T.U und wurde mit LyX 1.3.4 erstellt.

### 9.1 Danksagung

Am Ende gibt es stets Personen, ohne die die Arbeit nicht möglich gewesen wäre. Es sind vor allem

- Dietmar Bär <dietmar.baer@atu.de>, der als unser „Guru“ schon im Vorfeld immer wieder kritische Lücken gefüllt und Anregungen gegeben hat
- Stefan Pongratz <stefan.pongratz@atu.de>, Christian Argauer <christian.argauer@atu.de> und Michael Käck <michael.kaeck@atu.de>, die uns wochenlang ertragen mußten und trotzdem noch Zeit zum Lektorat fanden
- Jan Schampera <jan.schampera@unix.net>, der vielerlei zur Verwendung von LyX und der äußeren Form beigetragen hat.

Ihnen gilt, neben allen anderen Helfern, unser besonderer Dank.

## 9.2 Autoren

**Martin Klier** ist Systemadministrator für Server auf Linux-Basis und Datenbankadministrator für Oracle 10g-Systeme in der A.T.U-Zentrale in Weiden.

**Martin Blattner** arbeitete im selben Bereich als Entwickler und DBA für Oracle 9i- und 10g-Database-Systeme und ist zwischenzeitlich für Swisscom tätig.

Veröffentlichung genehmigt: Manfred Gerlach, EDV- und Organisationsleiter A.T.U

## Index

- Basis, zertifizierte, 3
- Bibliotheken, Oracle, 30
- bonding, HowTo, 8
- CacheFusion, 8
- CRS - Cluster Ready Services, 15
- crsd, 19
- dbca, 24, 26
- Dienste, Startreihenfolge, 19
- Diskgroup, 25
- DNS, 9
- evmd, 19
- Failgroup, 25
- Failover, clientseitiger, 29
- Fehlerfall, Hostausfall, 32
- gkrellm - System Monitoring, 9
- hosts, 9
- initfile, 34
- Installationssatz, CRS, 15
- Installationssatz, RDBMS, 22
- Instanzen, Zusammenhang der, 24
- Libraries, Oracle, 30
- Listener, 30
- listener.ora, 33
- Loadbalancing, serverseitiges, 29
- lsnrfg, 30
- Multipath, 12
- Note 259301.1, 19
- Note 263897.1, 19
- Note 265769.1, 19
- Note 293988.1, 7
- ocssd, 19
- orarun.rpm, 4
- pfile, 34
- raw device, 13
- RDBMS, Installation, 22
- Reihenfolge d. Dienste, 19
- spfile, 25
- spfile, verändern, 34
- Split-Brain-Prevention, 17
- sqlplus, 30
- srvctl, 36
- ssh, Schlüsseltausch mit, 10
- ssh-agent, 10
- ssh-keygen, 10
- TAF, 29
- TCP-Verbindung, 32
- TNS-Protokoll, 30
- TNS-Verbindung, 32
- tnsnames.ora, 31, 36
- Volumegroup, 24
- Volumes, 11

## Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>1</b>
1.1	Hinweis! . . . . .	1
1.2	Ziel und Konzept . . . . .	2
1.3	Umfeld . . . . .	2
<b>2</b>	<b>Vorbereitung</b>	<b>3</b>
2.1	Global . . . . .	3
2.1.1	Pakete . . . . .	4
2.1.2	Kernel . . . . .	6
2.1.3	Zeichensatz . . . . .	7
2.2	Netzwerk . . . . .	7
2.2.1	Öffentliches Netz . . . . .	7
2.2.2	Interconnect mit NIC-bonding . . . . .	8
2.2.3	/etc/hosts . . . . .	9
2.2.4	ssh, Schlüsseltausch . . . . .	10
2.3	Volumes . . . . .	11
2.3.1	Hardwarezugriff . . . . .	11
2.3.2	Benötigte Volumes . . . . .	11
2.3.3	Multipath . . . . .	12
2.3.4	RAW-Devices . . . . .	13
2.4	Abschluß . . . . .	15
<b>3</b>	<b>Cluster-Ready-Services CRS</b>	<b>15</b>
3.1	Installation . . . . .	15
3.2	Abschluß . . . . .	18
3.2.1	Überprüfung . . . . .	18
3.2.2	Bootfähigkeit herstellen . . . . .	18
3.3	Starten und Stoppen . . . . .	19
3.4	Überprüfung und Tests . . . . .	20
3.4.1	WICHTIG . . . . .	20
3.4.2	VIPs . . . . .	21
3.4.3	Service Relocation . . . . .	21
3.4.4	Gesamtstatus . . . . .	22

---

<b>4</b>	<b>Installation Datenbank-System (RDBMS)</b>	<b>22</b>
<b>5</b>	<b>Instanzen</b>	<b>23</b>
5.1	Theorie . . . . .	24
5.1.1	ASM- und Nutzinstanz . . . . .	24
5.1.2	Diskgroup/Failgroup . . . . .	25
5.1.3	spfile . . . . .	25
5.1.4	undo-Tablespaces . . . . .	25
5.1.5	redo-Logs . . . . .	25
5.1.6	Cluster . . . . .	25
5.2	Instanzerzeugung mit dbca . . . . .	26
5.3	Überprüfung . . . . .	29
<b>6</b>	<b>Clusterfunktionalität</b>	<b>29</b>
6.1	Theorie . . . . .	29
6.1.1	Listener . . . . .	30
6.1.2	Clientzugriff . . . . .	30
6.1.3	VIPs und TAF . . . . .	32
6.2	Einrichten . . . . .	33
6.2.1	Server . . . . .	33
6.2.2	Client . . . . .	35
<b>7</b>	<b>Fail-Over, Recovery</b>	<b>36</b>
<b>8</b>	<b>Nützliche Kleinigkeiten</b>	<b>36</b>
8.1	srvctl . . . . .	36
8.1.1	Instanz . . . . .	37
8.1.2	ASM . . . . .	37
8.1.3	Ressourcen und Hilfsdienste . . . . .	38
8.2	Ressourcenverwaltung mit <i>crs_*</i> . . . . .	38
8.2.1	Status anzeigen mit <i>crs_stat</i> . . . . .	38
8.2.2	Manipulieren mit <i>crs_*</i> . . . . .	39
<b>9</b>	<b>Editorial</b>	<b>39</b>
9.1	Danksagung . . . . .	39
9.2	Autoren . . . . .	40

## List of Algorithms

1	/etc/sysconfig/oracle . . . . .	5
2	oracle.sh . . . . .	6
3	ifcfg-bond0 . . . . .	8
4	/etc/hosts . . . . .	9
5	/etc/sysconfig/suseconfig . . . . .	10
6	/etc/raw . . . . .	14
7	init.crs, Header . . . . .	19
8	tnsnames.ora für dedizierten Zugriff . . . . .	31
9	listener.ora node1 . . . . .	33
10	pfile TESTRAC . . . . .	34
11	tnsnames.ora für shared Zugriff . . . . .	36

## Tabellenverzeichnis

1	CRS Node Names . . . . .	16
2	CRS Netzwerkschnittstellen . . . . .	17
3	Instanzen - Namensvergabe . . . . .	24