Oracle Real Application Clusters 11gR2 Prozessarchitektur

Oracles *Real Application Cluster* (RAC) blickt auf eine gut 25jährige Entwicklungsgeschichte zurück. Von den ersten Anfängen als *Clustered Oracle* (1986), über die erste produktive Version als *Oracle Parallel Server* in Oracle 6 (1989), die Einführung von *Cache Fusion 1* in Oracle 8i (1991), der Umbenennung in *Real Application Clusters* in Oracle 9i (2001) und den Architekturumbrüchen in Oracle 10.1 (2004) und 11.2 (2009) hat RAC heute einen Komplexitätsgrad erreicht, der eine überschaubare Darstellung unmöglich macht. Die Prozesse in dieser Übersicht stellen eine Momentaufnahme auf einem RAC-Knoten mit Oracle Enterprise Linux 5.4 64bit in Oracle VM dar.

Ca. 120 Prozesse werden für den Betrieb eines RAC auf jedem Knoten gestartet. Es wurde Gridinfrastrutkur 11.2.0.1 und eine Datenbank 11.2.0.1 Enterprise Edition installiert. Mit asmca wurde ein ASM Clusterdateisystem angelegt. Die Zahlen am linken Rand der Prozesse entsprechen der relativen Startreihenfolge. Innerhalb der ASM-Instanz und der Datenbank-Instanz wurde eine eigene Nummerierung vorgenommen, die innerhalb der Instanzen den Startreihenfolgen entspricht. Die Startreihenfolge kann leicht variieren: so kommt es u.a. vor, dass der OHASD-abhängige Rootagent (hier Prozessnummer Nr. 4), der meist direkt nach dem Oracle-Agenten (Nr. 3) startet, auch nach dem CSSD-Agenten (Nr. 10) starten kann. Die Färbung der Clustewareprozesse entspricht den in Metalink Note 153147.1 (*11gR2 Clusterware and Grid Home - What You Need to Know*) definierten Start-Level: Level 1=blau (OHASD, OHASD-Agenten, CSSD-Monitor); Level 2=rot (OHASD-Ressourcen, sichtbar mit `crsctl stat res -t -init`); Level 3 = grün (CRSD-Agenten); Level 4=violett (CRSD-Ressourcen, sichtbar mit `crsctl stat res -t`).

Thorsten Grebe, 24. November 2010

Diese Übersicht können Sie über oracle@science-computing.de beziehen oder unter www.science-computing.de/oracle herunterladen.

science + computing
A Bull Group Company

Legende
roter Rahmen → root-Prozesse
(r) → läuft mit respawn/restart-Option
Funkturm → kommuniziert mit Nachbarknoten
Blitz → Prozessende erzwingt Reboot
→ Kommunikation
→ I/O (Lese/Schreibzugriffe)

OHASD-abhängiger Root-Agent läuft mit ca. 20 Threads. Er startet die Prozesse für den ACFS-Zugriff, startet den CRSD, den Disk-Monitor und den neuen Zeitservice CTSS. Der Agent kann nicht einzeln gestoppt werden, läuft mit reboot-Flag.
Tool: `crsctl {start|stop|check} crs` (stoppt alle Prozesse ausser init.ohasd)
Logs: `$GRID_HOME/log/<Hostname>/agent/ohasd/orarootagent_root/`

OHASD-Init-Prozess ist der Mutterprozess aller Clusterprozesse.
Dieser Prozess lässt sich nicht beenden (gestartet mit respawn-Option). Er ist der einzige, der nach `"crsctl stop crs"` oder nach `"/etc/init.d/ohasd stop"` noch läuft. Er löst ab 11gR2 die drei ehemaligen Init-Skripte für CSS, EVM und CRS ab.
Log: `/var/log/messages`

Oracle High Availability Services Daemon läuft mit über 40 ohasd.bin-Threads. Er startet drei Agenten und den CSS-Monitor. Der ohasd-Prozess pflegt die lokale Cluster Registry (OLR).
Tool: `crsctl {start|stop|check} crs`
`/etc/init.d/ohasd {start|stop}`
Log: `$GRID_HOME/log/<Hostname>/ohasd/ohasd.log`

OHASD-abhängiger Oracle-Agent startet gipcd, mdnsd, gpnpd, evmd und die ASM-Instanz.
Tool: `crsctl {start|stop|check} crs`
Logs: `$GRID_HOME/log/<Hostname>/agent/ohasd/oraagent_oracle/`

Grid Interprocess Communication Daemon
Unterstützt Kommunikation über Interconnect. Ca. 2 Threads.
Log: `$GRID_HOME/log/<Hostname>/gipcd/gipcd.log`

Der ***Oracle Kernel Service***-Prozess bleibt beim Shutdown der Clusterprozesse solange bestehen, wie die ACFS-Module noch geladen sind. Insgesamt werden vom OHASD die vier Module oracleacfs, oracleadvm, oracleoks und oracleasm geladen. Die Kernelmodule *oracleacfs und oracleadvm* sind abhängig von *oracleoks*. Das Kommando `crsctl stop crs` entlädt sie komplett, es sei denn, es gibt hängende ACFS-Zugriffe. Dies erkennt man daran, dass nach dem `crsctl stop crs`-Kommando neben dem OHASD-init Prozess noch der oks_wkq Prozess besteht. Die ACFS-Treiber vermitteln den Zugriff auf die im ASM liegenden *Dynamic Volumes* (=ADVM), die als Blockgerätedateien unter /dev/asm/ exportiert werden. Die CRS-Ressource `ora.registry.acfs` prüft alle 30 Sekunden, ob die ACFS-Mounts noch bestehen.
Tool: `lsmod | grep oracle`
`acfsutil; /etc/init.d/oracleasm`
`srvctl` (z.B. *`stop filesystem -d /dev/asm/daten101`*)
Log: `/var/log/messages`
`acfsutil log -f /tmp/acfs.log` *Metalink 885363.1*
`event="15199 trace name context forever, level 2097159"`

CSS Der ***Cluster Synchronization Service*** besteht aus drei CSS-Prozessen und einem verlängerten Arm: Die drei Prozesse sind der CSS-Monitor, der CSS-Agent und der eigentliche CSS-Daemon. Der verlängerte Arm sind zwei Kill-Daemon.
Die Aufgabe des CSS ist die Kontrolle der Knotenmitgliedschaften im Cluster.
CSS entscheidet, welche Knoten am Clusterverbund teilnehmen bzw. ausgeschlossen werden. Er prüft den Status von ASM- und Datenbankinstanz und hilft dadurch, die Notwendigkeit für ein Recovery zu erkennen. CSS informiert alle Clusterknoten über EVM und ONS über Statusänderungen, z.B. den Aus- oder Eintritt eines Clustermitglieds. Die drei CSS-Prozesse laufen mit erhöhter Priorität (139) in der *Fixed Priority Class*. Sie starten jeweils ca. 15-25 Threads. CSS wird auch in einer Einzelinstanz benötigt, wenn ASM im Einsatz ist.
Tool: `crsctl {get|check} css`

Der ***CSSD-Agent*** startet und stoppt den eigentlichen CSSD, den ocssd.bin-Prozess, und prüft, ob dieser hängt. In einem solchen Fall erzwingt er einen sofortigen Reboot des Knoten. Der CSSD-Agent prüft auf hängende Nachbarknoten, übernimmt I/O-Fencing, indem hängende Knoten neu gestartet werden. Zusammen mit dem CSSD-Monitor löst der CSSD-Agent den ehemaligen Prozess `OPROCD` und das Modul `hangcheck-timer` ab, die bis 11.1. Hängezustände überwachten. Nach wieviel Sekunden ein Hängezustand diagnostiziert wird, kann über `crsctl get css misscount` ermittelt werden (Vorgabe 30 s). Ein Kill auf `cssdagent.bin` führt zum sofortigen Nachstarten des Prozesses durch OHASD. Der CSSD-Agent läuft unter root mit ca. 20 Threads. Vermutlich startet der CSSD-Agent die Kill Daemon.
Tool: `crsctl` (Der Agent kann nicht einzeln angesprochen werden.)
Logs: `$GRID_HOME/log/<Hostname>/agent/ohasd/oracssdagent_root/`

Der ***CSSD-Monitor*** prüft, wie cssdagent auch, auf hängende Knoten (lokal und remote) durch das Überwachen des Prozess-*Schedulings*. Wie der Agent, kann auch der Monitor bei Entdecken von Hängezuständen den sofortigen Neustart eines Clusterknotens erzwingen. Läuft unter Root mit ca. 20 Threads.
Logs: `$GRID_HOME/log/<Hostname>/agent/ohasd/oracssdmonitor_root/`

Der eigentliche ***Cluster Synchronization Service Daemon*** prüft den *Heartbeat* zu Nachbarknoten und den Zugriff auf die *Voting Disks*; leitet *Node Evictions* ein; kann über IPMI Nachbarknoten herunterfahren (*Node-termination Escalation*). Ist das ocssd.bin-Prozesses führt zum sofortigen Reboot. Der erfolgreiche Start von ocssd.bin erfordert, dass GPnP-Dienst gestartet ist, auf Profile zugreifen kann und alle *Voting Disks* identifiziert werden können. CSSD kommuniziert mit EVM, CRSD und Clusterware von Fremdanbietern. Läuft mit über 20 Threads. Liegen die *Voting Disks* im ASM, wird deren Lokation mit `$GRID_HOME/bin/kfed` ausgelesen (ASM-Instanz startet nach CSS).
Tool: `crsctl {query|add|delete} css votedisk` `crsctl {start|stop}-res ora.esd -init`
Logs: `$GRID_HOME/log/<Hostname>/cssd/`

frühe ACFS-Prozesse

Errorlogs werden erstellt unter:
`$GRID_HOME/log/<Hostname>/acfs*/`

acfs_bast0 acfs_bast4
acfs_bast1 acfs_bast5
acfs_bast2 acfs_bast6
acfs_bast3 acfs_bast7

8 ***acfs_bast***-Prozesse werden gestartet. Die Funktion ist nicht dokumentiert. Die Prozesse laufen unter root und mit erhöhter Priorität 34. Um Trace-Information zu erhalten muss ein Event gesetzt werden (Metalink 885363.1) oder der Loglevel mit `"acfsutil log"` gesetzt werden.
Log: `/var/log/messages`
`$GRID_HOME/log/<Hostname>/acfs*/`

Cluster Ready Services Daemon
ist der zentrale Dienst für die Verwaltung von Ressourcen im Cluster. CRS pflegt die Oracle Cluster Registry (OCR) und stoppt, startet oder verschiebt Ressourcen so, wie es in der OCR konfiguriert ist oder mit `srvctl` vom DBA geführt wird. Zu den Ressourcen des CRS zählen Listener, Datenbank-Instanz und Services. Der CRSD-Prozess erzeugt *Events*, wenn sich der Status von Ressourcen ändert. Diese *Events* werden vom EVM veröffentlicht. Bis Version 11.1 wurde der CRSD über ein init-Skript gestartet, ab 11.2 ist er abhängig vom OHASD-Rootagent. Die Verwaltung des CRSD erfolgt über `srvctl` und nicht, wie der Name suggeriert, über `crsctl`. Der CRSD-Prozess läuft unter Root mit der Reboot-Option und startet über 50 Threads.
Tool: `srvctl {stop|start|...} resource`
Log: `$GRID_HOME/log/<Hostname>/crsd/crsd.log`

CRSD-abhängiger Rootagent startet GNS, Netzwerk-Ressourcen, führt ACFS-Mounts durch (ruft dazu `acfsregistrymount` auf). Der Agent wird indirekt über den CRSD verwaltet, der DBA kann ihn nicht direkt ansprechen.
Logs: `$GRID_HOME/log/<Hostname>/agent/crsd/orarootagent_root/`

CRS

Grid Naming Service
Optionaler Namensdienst der Clusterware. Bezieht alle IP-Adressen für den Cluster vom externen DHCP. GNS ist mit einer eigenen virtuellen IP-Adresse beim zentralen DNS registriert und läuft als Ressource immer nur auf einem Knoten. GPnP kontaktiert GNS zur Lokalisierung von Profilen. GNS leitet Anfragen an mDNSD weiter, fungiert als Gateway zwischen externem DHCP und mDNSD. Übernimmt alle DNS-Anfragen an externe DNS-Server (*subdomain delegation*).
Tool: `srvctl{config|...} gns`
Log: `$GRID_HOME/log/<Hostname>/gnsd/gnsd.log`

ACFS-Prozesse laufen unter Root. `acfssnap0` wird gestartet, wenn mit acfsutil ein Snapshot erstellt wird. Dabei werden Änderungen in ein verstecktes Verzeichnis `.ACFS` kopiert und in der View `V$ASM_ACFSSNAPSHOTS` protokolliert. `acfsvol/acfsds` sind nicht dokumentiert.
Tool: `acfsutil`
Logs: `/var/log/messages`
`$GRID_HOME/log/<Hostname>/acfs*/`

Oracle Kernel Services-Prozess, kann einfach oder mehrfach gestartet sein – später Prozess; läuft mit erhöhter Priorität 33. Bedeutung ist nicht dokumentiert.
Log: `/var/log/messages`

Cluster Time Synchronization Service *CTSS*
Der ***Cluster Time Synchronization Service*** achtet darauf, dass die Uhrzeit unterhalb der Clusterknoten synchron gehalten wird. Er kann im passiven Observer-Modus laufen, wenn NTP aktiv ist (empfohlen). Bei fehlender oder defekter NTP-Konfiguration wechselt CTSS automatisch in den aktiven Modus und synchronisiert die Zeiten der Clusterknoten untereinander. CTSS bremst und beschleunigt dazu die Uhren, setzt aber niemals eine Uhrzeit zurück. Ein Knoten wird automatisch als Referenz gewählt, es wird nicht gegen eine externe Uhr synchronisiert. Stoppen und starten von NTP ändern den CTSSD-Modus. Mit `crsctl` kann der Modus geprüft, aber nicht geändert werden.
Tool: `crsctl check ctss`
Log: `$GRID_HOME/log/<Hostname>/ctssd/octssd.log`

CRSD-abhängiger Oracleagent startet den *Oracle Notification Service*, die Listener und schließlich die Datenbankinstanz. Er führt *Server Callout* Skripte aus, wenn FAN-Events empfangen werden. Falls 9i Datenbanken im Cluster verwaltet werden müssen startet der Oracleagent den GSD (*Global Service Daemon*, hier nicht gezeigt), dieser ist in der Standardinstallation deaktiviert. Oraagent übernimmt die Rolle der ehemaligen RACG-Prozesse (racgmain, racgimon). Der Prozess läuft mit ca. 20 Threads. Wurden Datenbank und Grid-Infrastruktur mit unterschiedlichen Benutzern angelegt, wird ein zusätzlicher oraagent gestartet, dessen Aufgabe es ist, die Datenbank-Instanz zu starten.
Logs: `$GRID_HOME/log/<Hostname>/agent/crsd/oraagent_oracle/`

Die beiden ***OCLKD-Kill Daemon*** sind der verlängerte Arm des CSS. Ein Kill Daemon läuft mit den Rechten des Oracle Softwarebesitzers, der andere als Root. Der CSS-Dienst verwendet die beiden *Kill Daemon*, um Instanzen zu beenden, für die eine Stoppanfrage von anderen Knoten empfangen worden ist. Welcher Prozess die beiden Killdaemon tatsächlich startet ist nicht dokumentiert. Vermutlich stehen sie unter Kontrolle des cssdagent- oder des cssdmonitor-Prozesses. Ab 11.2.0.2 laufen die Kill Daemon nicht mehr als Prozesse, sondern als Threads innerhalb der CSS-Prozesse.
Log: `$GRID_HOME/log/<Hostname>/client/oclskd.log`

Grid Plug & Play Daemon hilft bei der automatischen Bereitstellung von neuen Knoten im Cluster. Knotenname und Netzwerkkonfiguration werden automatisch erstellt. GPNP ist nicht für das Hinzufügen von Instanzen zuständig (hierzu dienen Serverpools). GPNP verwendet den Zeitdienst CTSS, den Clusternamensdienst GNS, die SCAN-Adressen und die Serverpool-Funktion. Zum Auflösen von Knotennamen wendet er sich an den mDNS. Diese Dienste (inklusive GNS) müssen für GPNP gestartet sein. Knotenkonfigurationen speichert GPNP als Profile in XML-Dateien: `$GRID_HOME/gpnp/profiles/peer/profile.xml`. Der Inhalt kann über `gpnptool get` ausgelesen werden. Falls manuelle Änderungen an Profildateien vorgenommen werden, müssen diese vor und nach der Änderung mit `gpnptool` ensigniert und schließlich wieder signiert werden. In der Regel werden Profildateien automatisch über `oifcfg` aktualisiert. Die Profile enthalten u.a. die Clusterid und die Lage der ASM-Disks. Der Prozess läuft mit ca. 7 Threads.
Tools: `oifcfg, gpnptool`
Log: `$GRID_HOME/log/<Hostname>/gpnpd/gpnpd.log`

Multicast Domain Name Service beantwortet DNS-Anfragen im Auftrag von GNS; entdeckt Services innerhalb des Clusters; startet mit ca. 3 Threads.
Log:`$GRID_HOME/log/<Hostname>/mdnsd/mdnsd.log`

Event Manager

Der ***Event Manager*** empfängt Nachrichten vom CSSD und CRS und publiziert sie über ONS an Abonnenten. Er kommuniziert mit den Nachbarknoten im Cluster. Evmd.bin scanned das Callout-Verzeichnis `$GRID_HOME/racg/usrco` und ruft Callout-Skripte auf.
Konfiguration: `$GRID_HOME/evm/admin/conf/evmlogger.conf`
Logs: `$GRID_HOME/evm/log/<Hostname>_evmdaemon.log` `$GRID_HOME/log/<Hostname>/evmd/evmd.log`

Der ***EVM-Logger*** wird für rückwärtige Kompatibilität benötigt. Er liest die Konfiguration des Event Managers aus und verteilt benutzerdefinierte Ereignisse. Der Start des EVM-Loggers kann verzögert erfolgen.

Oracle Notification Service besteht aus zwei ONS-Prozessen: Der erste ist der Mutterprozess, der den eigentlichen ONS-Prozess startet und überwacht. Die ONS-Prozesse werden nicht aus $GRID_HOME/bin sondern aus $GRID_HOME/opmn/bin gestartet. ONS handhabt zwei Arten von Events: FAN (*Fast Application Notification*)-Events (oder HA-Events) und RLB (*Runtime Load Balancing*)- Events. FAN-Events werden an abonnierende Anwendungen und Prozesse geleitet. Dazu empfängt ONS eine Untermenge der veröffentlichten Clusterware Events vom lokalen EVM. Fast Connection Failover (FCF) verwendet die vom ONS versendeten Events. Ebenso verwendet *Runtime Load Balancing Advisory* Information, die vom ONS gesendet werden. ONS empfängt Last- und Advisordaten vom MMON der Datenbankinstanz und leitet die Information an die lokalen Listener und an Anwendungen weiter. Die beiden ONS-Prozesse werden unterstützt von dem eONS-Javaprozess (s.Prozess Nr. 29). Der ONS-Prozess bildet ca. 15 Threads.
Konfigurationsdateien: `$GRID_HOME/opmn/conf/ons.config` und `opmn.xml`
Debug-Anleitung in Metalink 759895.1: *"The ONS Daemon Explained in RAC/CRS Environment"*
Tool: `srvctl {config|modify|enable|disable} ons` `onsctl {start|stop|ping|debug}`
Logs: `$GRID_HOME/opmn/logs/ons.log.<Hostname>`

Der SCAN-Listener horcht auf einer virtuellen IP-Adresse, die vom GRS vergeben wird (ohne GNS müssen für den SCAN-Listener drei IP-Adressen beim zentralen DNS-Server hinterlegt werden). Client-Anwendungen sollten gegen den SCAN-Listeners gerichtet werden. In der Datenbankinstanz wird der Parameter REMOTE_LISTENER auf die SCAN-Adresse gesetzt, damit sich die Datenbankservices dort registrieren.
Tool: `srvctl {config|start|stop|relocate|remove|modify|enable|disable} scan_listener`
Logs: `$GRID_HOME/opmn/logs/ons.log.<Hostname>`

Enhanced Oracle Notification Service ist ein Java-Prozess der den ONS (Prozess Nr. 26) funktional erweitert. Startet mit ca. 40 Threads.
Tool: `srvctl {config|start|stop} eons`
Logs: `$GRID_HOME/log/<Hostname>/srvm/`

Disk-Monitor *Disk-Monitor*
Der ***Disk-Monitor*** ist für I/O Fencing in einer Exadata-Konfiguration verantwortlich. Da Exadata zu jeder Zeit zu einem RAC hinzugefügt werden können soll, läuft dieser Dienst immer, sobald der CSS-Dienst startet.
Log: `$GRID_HOME/log/<Hostname>/diskmon/diskmon.log`

Storage

ACFS-Verzeichnisse /dev/asm/
OLR **Redo** **Undo** **Voting Disk** **Daten** **OCR**

Jeder Clusterknoten verwaltet seine eigene *Local Registry* (OLR), Redo und Undo. Redo wird vom Logwriter (LGWR) geschrieben, Undo vom Serverprozess. Für die Pflege der OLR ist der lokale OHASD-Prozess verantwortlich.

Die *Oracle Cluster Registry* (OCR), die Voting Disks und die eigentlichen Datendateien müssen von allen Knoten gelesen und beschrieben werden. Integritätscheck der OCR/ OLR mit `ocrcheck [-local]` oder `cluvfy comp [ocr|olr]`

ASM-Instanz

1 **PMON** ***Process Monitor*** s. Beschreibung bei DB-Instanz
2 **VKTM** ***Virtual Keeper of Time*** s. Beschreibung bei DB-Instanz
3 **GEN0** ***General Task Execution Process*** s. Beschreibung bei DB-Instanz
4 **DIAG** ***Diagnosability Process*** s. Beschreibung bei DB-Instanz
5 **PING** ***Ping Process*** s. Beschreibung bei DB-Instanz
6 **PSP0** ***Process Spawner*** s. Beschreibung bei DB-Instanz
7 **DIA0** Helferprozess des ***Diagnosability Process*** *s. Beschreibung bei DB-Instanz*
8 **LMON** **Lock Monitor** undokumentiert, was dieser Prozess in einer ASM-Instanz macht
9 **LMD0** **Lock Manager Daemon** undokumentiert, was dieser Prozess in einer ASM-Instanz macht
10 **LMS0** **Lock Manager Server** Undokumentiert, was dieser Prozess in einer ASM-Instanz macht. Wie der LMS-Prozess der Datenbankinstanz läuft auch der ASM-LMS0 mit erhöhter Priorität 41 ist der *Fixed Priority Class*.
11 **LMHB** ***LM Heartbeat Monitor*** überwacht LMON, LMD, LMS
12 **PMAN** ***Memory Manager*** s. Beschreibung bei DB-Instanz
13 **DBW0** ***Database Writer*** verwaltet den SGA Buffer Cache der ASM-Instanz. Schreibt auch hier schmutzige Blöcke geänderter ASM-Metadaten auf die Festplatte.
14 **LGWR** **Log Writer** pflegt die Buffer des *ASM Acitve Change Directory* (ACD) und schreibt Anderungen auf die Festplatte
15 **CKPT** **Checkpoint Process** verwaltet im ASM Cross-Instance Calls
16 **SMON** ***System Monitor*** steht in Kontakt mit CSS zur Überwachung des Systemzustands.
17 **RBAL** ***Rebalance Process*** Der *Rebalance Master* koordiniert Extent-Umverteilungen, wenn Disks zur ASM-Instanz hinzugefügt oder entfernt werden. Die tatsächlich Rebalancetätigkeit wird von ARBn ausgeübt (hier nicht gestartet). Ein gleichnamiger Prozess RBAL existiert auch in der DB-Instanz, hat dort jedoch eine andere Aufgabe.
18 **GMON** ***Diskgroup Monitor*** koordiniert Cluster- und Gruppenmitgliedschaft innerhalb der ASM-Instanz. Verantwortlich für Drop/Offline-Aktionen. Erzeugt Bnnn-Prozesse, wenn Offline-Disk entdeckt wird, mit denen der Offline-Status überwacht wird.
19 **MMON** ***Manageability Monitor Process*** s. Beschreibung bei DB-Instanz
20 **MMNL** ***Manageability Monitor Light*** s. Beschreibung bei DB-Instanz
21 **LCK0** **Lock Prozess** undokumentiert, was dieser DB-Prozess in einer ASM-Instanz macht
22 **VBG0** ***Volumne Background Process 0*** wird für den Zugriff auf ACFS-Verzeichnise benötigt. Verwaltet Anfragen vom ADVM-Treiber des Betriebssystems. Öffnet und schließt ADVM-Volumes, z.B. beim Mount oder Umount.
23 **ACFS** ***ASM-Cluster Filesystem Prozess*** Der ACFS-Hintergrundprozess koordiniert Cluster Mitgliedschaft und Gruppenmitgliedschaft innerhalb der ASM-Instanz. Protokolliert, welche Konten am Cluster teilnehmen.
24 **ASM1_ocr** ***OCR-Prozess*** wird gestartet, wenn die *Cluster Registry* (OCR) im ASM liegt, was in 11.1 nicht möglich war und ab 11.2 als empfohlene *Best Practice* gilt.
25 **ASMB** ***ASM Background Process*** und ***Bequeth-Verbindung*** zur ASM-Instanz selbst. ASMB kontaktiert CSS, um ASM-Connectstrings für Diskgruppen zu erfahren. Über die Bequeth-Verbindung werden Heartbeat- und Statistik-Informationen bezogen. Beide Prozesse, ASMB und die Bequeth-Verbindung, existieren auch in der DB-Instanz.
26 **ASM1_asmb_+asm1** ***Volume Driver Background Process*** unterstützt Zugriff auf ACFS-Verzeichnisse. Gibt ASM-Kommandos gegen Diskgroups an ADVM-Treiber weiter
27 **VDBG** ***Volume Member Background Process*** implementiert I/O Fencing. Er stellt sicher, dass ASM-Instanzen nicht durchgeführt wird, bevor alle ADMV I/O-Vorgänge abgeschlossen sind. Führt zusammen mit VDBG I/O-Zugriffe auf ACFS-Verzeichnisse aus.
28 **VMB0** ***Volume Background Process 1-9*** unterstützen die Tätigkeit von VBG0
29 **VBG1**
30 **VBG2**

Die **ASM-Instanz** wird vor dem CRS-Daemon gestartet, aber erst nach dem CSS. Liegen die *Voting Disks* im ASM, muss der CSS deshalb in der Lage sein, ohne laufende ASM-Instanz direkt auf diese zugreifen zu können.
Die rot unterlegten Prozesse PING2 und L* werden nur in einem RAC aktiviert. Die Funktion der L*-Prozesse in der ASM-Instanz ist nicht dokumentiert. LMON, LMD, LMS und LCK sind eigentlich Dienste der Datenbankinstanz, die das Block-Shipping über den Interconnect regulieren. Die grünen Prozesse stellen Standardprozesse einer Datenbank-Instanz dar. Auch hier ist die Funktion in der Datenbankinstanz häufig undokumentiert, wie z.B. für DBWR und LGWR. Ab 11.1 werden alle Logs, Traces und Dumps im *Automatic Diagnostic Repository* **(ADR)** gesammelt. Ausser für Rebalance-Operation führt die ASM-Instanz in der Regel kein I/O für die Datenbank-Instanz aus. Die Serverprozesse der DB-Instanz greifen direkt auf die ASM-Datendateien zu.
Logs: `$ORACLE_BASE/diag/asm/+asm/+ASM<n>/trace`

ASM-Legende
grün → Standardprozess
gelb → nur in ASM-Instanz
rosa → ASM-Funktion
rot → nur bei RAC
<00> → Startreihenfolge

Datenbank-Instanz

Zusammen pflegen GCS und GES das *Global Resource Directory* (**GRD**), ein in-Memory Register aller im Cluster gehelten Block- und Cache-Ressourcen, das für alle Instanzen sichtbar ist. Bei einem Knotenausfall wird das GRD kurz eingefroren und durch LMON und LMS aktualisiert. Während dieser sogenannten Remastering-Periode ist die Datenbank für Benutzer nicht verfügbar.

GCS Die **Global *Cache Services*** bestehen aus dem LMSn-Prozess. Er verfolgt den Status und den Aufenthaltsort von Datenblöcken. GCS sendet Blöcke über den Interconnect an anfragende Instanzen.

GES Die ***Global Enqueue Service*s** werden von den drei Prozessen LMON, LMD und LCK gebildet. GES verwaltet globale Sperren für Ressourcen im Cluster. Dies können neben Datenblöcken auch Ressourcen im *Dictionary-* und *Library-Cache* sein. Für nicht-Datenblock-Ressourcen ist LCK zuständig. GES ist auch für das Erkennen von Deadlocks im Cluster verantwortlich.

1 **PMON** ***Process Monitor*** Überwacht andere Hintergrundprozesse. Beendet bei Bedarf die Instanz. Räumt abgebrochene Sitzungen auf. Registriert Instanz beim Listener. Immer der erste Prozess.
2 ***VKTM*** ***Virtual Keeper of Time*** Interner hoch auflösender Timer zur präzisen Messung von Zeitintervallen. Läuft mit erhöhter Priorität 41 ist der *Fixed Priority Class* (entspricht *Real Time*).
3 ***GEN0*** ***General Task Execution Process*** entlastet andere Prozesse, indem er langlaufende Wartungsjobs übernimmt, z.B. ASM-Diskoperationen.
4 ***DIAG*** ***Diagnosability Process*** Vor 11 nur in RAC, ab 11 Standardprozess. Überwacht Gesamtstatus der Instanz. Erstellt diagnostische Dumps, führt globale oradebug-Kommandos aus. Wird unterstützt von DIA0.
5 ***DBRM*** ***Database Resource Manager Process*** setzt Ressourcenpläne des *Resource Managers* um.
6 ***PING*** ***Ping Process*** misst Netzwerklatenz, indem er Testnachrichten über den Interconnect an die Nachbarinstanzen sendet und die Zeit bis zur Empfangsbestätigung protokolliert.
7 ***PSP0*** ***Process Spawner*** startet andere Prozesse, vor allem während Instanzstart
8 ***ACMS*** ***Atomic Controlfile to Memory Server*** stellt sicher, das clusterweite Aktualisierungen der SGAs transaktionssicher (atomar) durchgeführt werden
9 ***DIA0*** Helferprozess des ***Diagnosability Process.*** Verantwortlich für das Erkennen von Hängezuständen und für das Auflösen von Deadlocks.
10 ***LMON*** ***Global Enqueue Service Monitor (=Lock Monitor)*** überwacht Status aller globalen Queues und Ressourcen. Führt Lock-Recovery durch, wenn Instanzen den Cluster unerwartet verlassen. Es gibt immer nur einen LMON pro Instanz.
11 ***LMD0*** ***Global Enqueue Service Daemon (=Lock Manager Daemon)*** vermittelt Blockanfragen von LMS und hält Blöcke konsistent unterhalb der Instanzen. Bereinigt Deadlocks. Der lokal LMD0 kommuniziert mit dem LMD0 anderer Instanzen. Nur ein LMD wird pro Instanz gestartet
12 ***LMS0*** ***Global Cache Service Process (=Lock Manager Server)*** sendet Datenblocke an anfragende Nachbarinstanzen. Verwaltet Status von Datenblöcken. Bis zu 34 LMS-Prozesse, abhängig vom Init-Parameter `cpu_count`. Verantwortlich für Rollback von Blocken für anfragende Instanzen. Läuft mit erhöhter Priorität 41 ist *Fixed Priority Class* (entspricht *Real Time*).
13 ***RMS0*** ***Rac Management Server*** Hilfsprozess beim Verwalten von Ressourcen, dazu zählt das Erzeugen von Ressourcen, wenn neue Instanzen zum Cluster hinzukommen. Es gibt nur einen RMS Prozess pro Instanz
14 **LMHB** ***LM Heartbeat Monitor*** überwacht LMON, LMD, LMS
15 ***MMAN*** ***Memory Manager*** sammelt Statistiken über die Verwendung der SGA und passt die Größe von SGA-Komponenten an.
16 **DBW0** ***Database Writer*** Schreibt "schmutzige" Datenblöcke in Datendateien.
17 **LGWR** ***Log Writer*** Schreibt Redologs.
18 ***CKPT*** ***Checkpoint Process*** Aktualisiert Dateiheader bei Checkpoints.
19 **SMON** ***System Monitor*** Führt Transaktions- und Instanz-Recovery durch. Gibt Temp- und Undo-Segmente frei.
20 ***RECO*** ***Recovery Process*** Bereinigt nicht abgeschlossene 2PC-Transaktionen.
21 ***RBAL*** ***Rebalance Process*** RBAL öffnet die Disks in allen Diskgruppen, damit sie für Clients verfügbar sind. In der ASM-Instanz gibt es einen gleichnamigen Prozess, der dort andere Aufgaben erledigt.
22 ***ASMB*** ***ASM Background Process*** erhält Heartbeat zur ASM-Instanz und übermittelt dieser I/O-Statistiken. ASMB kontaktiert CSS, um den Connectstring für die ASM-Instanz zu erfahren. Über den Connecstring wird eine Bequeth-Verbindung zur Datenbank aufgebaut. Wenn ein Serverprozess eine neue Datei erstellt, sendet die ASM-Instanz die aktualiserte Extentmap an ASMB zurück.
23 ***MMON*** ***Manageability Monitor Process*** erhebt alle 60 Sek. Statistiken direkt aus der SGA und sendet Last- und Advisory-Daten über ONS an Listener, befüllt die Tabellen des AWR. MMON-Events können ebenso über die View `SYS$SERVICE_METRICS_TAB`.
24 ***MMNL*** ***Manageability Monitor Process Light*** unterstützt *MMON*; erhebt sekündlich Servicestatistiken für *Active Session History*, die aus `gv$servicemetric`, `gv$servicemetric_history` oder `v$active_session_history` ausgelesen werden können
25 ***D000*** ***Dispatcher Process***
26 **ASM1_asmb_orcl1** ***Bequeth Verbindung zur ASM-Instanz.*** Nachdem ASMB vom CSS den ASM-Connectstring erfahren hat, wird eine direkte Verbindung zur ASM-Instanz aufgebaut. Über diese Dauerverbindung werden Heartbeats und Statistiken ausgetauscht.
27 ***S000*** ***Shared Server Process***
28 ***MARK*** ***Mark Allocation Unit for Resync Koordinator*** aktualisiert *Staliness Register* für Extents, wenn ASM-Disks offline gehen. Die Information wird für die Resynchronisierung benötigt, sobald Disks wieder online sind.
29 ***LCK0*** ***Instance Enqeue Process (=Lock Process)*** koordiniert Anfragen für globale Ressourcen, die nicht Datenblöcke sind, z.B. Library- oder Row Cache-Anfragen. Ergänzt damit die Tätigkeit von LMD. LCK setzt Sperren für benötigte Ressourcen.
30 ***RSMN*** ***Remote Slave Monitor*** ist verantwortlich für das Erzeugen von Slave Hintergrundprozessen in Nachbarinstanzen und Kommunikation mit diesen.
31 ***ARC0*** 32 ***ARC1*** 33 ***ARC2*** 34 ***ARC3*** ***Archivierungs-Prozesse 0-3*** archivieren Redologs. Bis zu 30 Archiver-prozesse (ACR0-ARC29) können gestartet werden. Geschriebene Archivelogs können über `v$archived_log` ermittelt werden.
35 ***GTX0*** ***Global Transaction Process*** unterstützt im RAC globale Transaktionen. Neuer Prozess in 11g, der die manuelle Konfiguration von DTP-Diensten überflüssig macht. GLOBAL_TXN_PROCESSES muss > 0 sein (s. Metalink 462060.1). Bis zu 20 (GTX0-9, GTXa-j) werden gestartet, die Anzahl wird von Oracle automatisch festgelegt.
36 ***RCBG*** ***Result Cache Background Process***
37 ***QMNC*** ***AQ Coordinator*** beobachtet Eingänge in *Advanced Queues*. Startet **qnnn**-Prozesse. Anzahl der Prozesse wird über AQ_TM_PROCESSES kontrolliert.
38 ***Q000*** 39 ***Q001*** ***Advanced Queue Slave-Prozesse*** gestartet von QMNC
40 ***CJQ0*** ***Job Queue Coordinator*** beobachtet Job-Tabelle und startet Job Queue Prozesse **jnnn**. Für diese `JOB`-Aufträge. Anzahl wird durch `job_queue_processes` gedeckelt.
41 ***SMCO*** ***Space Mangement Coordinator*** *gehört zur Manageability Infrastructur. Koordiniert die Wiedergewinnung von Platz in Segmenten. Startet* **wnnn** *Slave-Prozesse.*
42 ***vkrm*** ***Virtual sKeduler for Resource Manager*** kontrolliert Prozessor-Scheduling für alle von *Resource Manager* verwalteten Prozesse.
43 ***w000*** 44 ***j000***

Datenbank-Legende
grün → Standardprozess
rosa → ASM-Funktion
blau → nur DB-Instanz
rot → nur bei RAC
<00> → Startreihenfolge

Die Datenbank-Instanz wird als letzte Komponente im RAC gestartet.
Ab 11.1 werden alle Logs, Traces und Dumps im *Automatic Diagnostic Repository* **(ADR)** gesammelt: `$ORACLE_BASE/diag/rdbms/<DB-Name>/<Instanz-Name>/`

Log- und Tracedateien

Gemeines Clusterware Alert Log:	`$GRID_HOME/log/<Hostname>/alert<Hostname>.log`
Letztes Lebenszeichen-Log:	`/etc/oracle/lastgasp/..`
ACFS-Treiber / ADVM-Module:	`/var/log/messages`
Logs der einzlenen Clusterwaredienste:	`$GRID_HOME/log/<Hostname>/<Komponenten-Name>/`
Logs der Agenten:	`$GRID_HOME/log/<Hostname>/agent/...` — *Unterverzeichnisse für die OHASD- bzw. CRSD-abhängigen Agenten*
Installationslogs:	`$GRID_HOME/cfgtoollogs/oui/`
OCJ4J:	`$GRID_HOME/log/<Hostname>/agent/crsd/ora_oc4j_type_oracle/...`
ONS:	`$GRID_HOME/opmn/logs/