Die Grid Infrastructure für Third-Party-Applications

Jörg Horchler, Bauer Systems KG

Schon seit der Einführung des RAC im Jahr 2001 wird von Oracle ein Cluster-Stack ausgeliefert. Ab Version 10.2 ist es mit der Clusterware möglich, Third-Party-Applications von diesem Stack mit selbst geschriebenen Skripten verwalten zu lassen. Die neue Version 11*g* R2 ist eine konsequente Weiterentwicklung. In diesem Artikel wird auf Neuerungen eingegangen und anhand eines Beispiels gezeigt, wie ein Server abgesichert werden kann.

In den Versionen 10.2 sowie 11.1 können mithilfe der „crs_*"-Befehle aus dem ORACLE_HOME der Clusterware externe Ressourcen angelegt, in der Oracle Cluster Registry (OCR) registriert und verwaltet werden. Die einzelnen Schritte sind:

- Erstellen eines Action-Skripts, das von der Clusterware aufgerufen wird.
- Erstellen eines Applikations-Profils mit dem Kommando „crs_profile"
- Registrieren des Profils mit dem Befehl „crs_register". Dadurch wird es in die OCR geschrieben
- Setzen von Berechtigungen mit dem Befehl „crs_setperm"
- Anschließend kann die Ressource verwaltet werden

In der Grid Infrastructure ist dieses Vorgehen im Prinzip nicht verändert. Allerdings wurden alle bisher verwendeten Kommandozeilen-Tools als veraltet markiert. Es sind nur noch zwei Befehle nötig:

- Der Cluster-Stack selbst sowie alle externen Ressourcen werden über den Befehl „crsctl" verwaltet
- Die Oracle-internen Ressourcen werden mit dem Befehl „srvctl" gesteuert. Der Name dieser Ressourcen beginnt immer mit der Zeichenfolge „ora"

Zusätzlich gibt es unter anderem noch folgende wichtige Unterschiede:

- Ein Action-Skript muss nun mindestens vier Aufrufe implementieren. Neben START, STOP und CHECK ist CLEAN hinzugekommen. Ferner gibt es noch den optionalen Einstiegspunkt „ABORT". CLEAN wird aufgerufen, wenn eine Ressource aufgeräumt werden muss. Bei einem Dateisystem würde dies einem „umount -f" entsprechen. Der ABORT-Aufruf hingegen wird immer dann ausgeführt, wenn einer der anderen Einstiegspunkte hängt, also nicht innerhalb des angegebenen Timeouts beendet wurde.
- Profile sind vollständig überflüssig. Stattdessen arbeitet die Grid Infrastructure komplett mit Ressource-Typen.

Architektur

Die Architektur der Grid Infrastructure unterscheidet sich stellenweise erheblich von der vorheriger Versionen. Gab es dort lediglich sieben Daemon-Prozesse, starten in der Grid Infrastructure bis zu 18. My Oracle Support [1] zeigt deren Start-Reihenfolge sowie die Funktionen der einzelnen Prozesse. Für die Verwaltung von 3rd-Party-Applikationen ist die wichtigste Änderung, dass die Clusterware jetzt mit Agents arbeitet. In den vorherigen Versionen gab es nur die Möglichkeit, Skripte oder Programme für die zu verwaltenden Ressourcen einzusetzen, die dann mit dem entsprechenden Entry-Point aufgerufen, abgearbeitet und beendet wurden. In der Grid Infrastructure ist es möglich, einen Agenten in C/C++ zu schreiben, der permanent online ist. Sind zum Beispiel sehr viele Filesysteme zu überwachen, kann der Overhead der Prozesserzeugung und der Ausführung von Skripten oder Programmen für jeden CHECK-Aufruf erheblich sein. In diesem Fall können ein Agent eingesetzt und die Filesysteme schneller gecheckt werden.

Es ist ebenfalls möglich, einen hybriden Agenten zu implementieren, bei dem einige Funktionen in C/C++ geschrieben und andere durch Skripte ausgeführt werden.

Beispiele für solche applikationsspezifischen und hybriden Agenten liegen im Verzeichnis „$ORA_CRS_HOME/crs/demo", die man mit einem installierten C++-Compiler übersetzen und ausprobieren kann. Anweisungen sind der Datei „readme.txt" zu entnehmen.

Ein weiterer Unterschied besteht darin, dass alle Ressourcen beim Start auf den Status „UNKNOWN" gesetzt werden und dann ein „CHECK" ausgeführt wird. Dadurch wird verhindert, dass eine Applikation ein Start-Kommando doppelt erhält und der Cluster eventuell den Start als fehlerhaft ansieht. Der Vorteil liegt auf der Hand: Will man eine Applikation nachträglich im laufenden Betrieb unter Cluster-Kontrolle bringen, muss man sie nicht als erstes herunterfahren, um sie dann starten zu lassen.

Ressource-Typen

In den vorherigen Versionen gibt es nur den Ressource-Typ „application", der praktisch nicht erweiterbar ist. Daher wird in der Praxis für jede Ressource ein eigenes Action-Skript erstellt, in dem die spezifischen Attribute gepflegt werden. Um Skripte generisch zu hal-

ten, kann man auch nicht genutzte Attribute des Application-Typs verwenden. Dieser Ansatz ist zum Beispiel in einem Whitepaper [2] verfolgt worden.

In der Grid Infrastructure ist dieses Konzept komplett überholt. Für den alten Typ „application“ gibt es einen eigenen Agenten mit Namen „appagent“, der aber nur zur Kompatibilität eingeführt wurde. Um Applikationen zu verwalten, die mit Skripten arbeiten, ist der neue „scriptagent“ zu verwenden. Mit dem Kommando „crsctl status type“ kann man sich alle bestehenden Typen und deren Basistypen anzeigen lassen. Alle Vorhandenen basieren dabei auf „resource“ und sind entsprechend erweitert. Ein Typ erbt dabei alle Attribute seines Basis-Typen, kann sie überschreiben und erweitern. Mit dem Befehl „crsctl status type <TYPE_NAME> -p“ kann man sich die speziellen Attribute anschauen.

Alle eigenen Applikationen sollten auf einem der beiden Typen „local_resource“ oder „cluster_resource“ basieren. Beide werden standardmäßig durch den „scriptagent“ verwaltet. Der Unterschied zwischen ihnen besteht darin, dass Ressourcen vom Typ „local_resource“ auf allen Clusterknoten gestartet werden. Wenn ein neuer Knoten in den Cluster aufgenommen wird, wird eine Ressource dieses Typs automatisch um den neuen Server erweitert und dort gestartet. Im Gegensatz dazu ist eine „cluster_resource“ eine Ressource, die ein- bis mehrmals im Cluster laufen und innerhalb der Maschinen wandern kann, wenn sie so konfiguriert wird.

Attribute

Die Attribute, die eine Ressource haben kann, sind in der neusten Version ebenfalls erweitert worden [3]. Die Wichtigsten sind:

- *ACTION_SCRIPT*
  Legt das auszuführende Skript fest
- *AGENT_FILENAME*
  Gibt den vollen Pfad zu dem Agenten-Programm an, dass für die Ressource verwendet werden soll. Jede Ressource muss von einem Agenten verwaltet werden
- *AUTO_START*
  Ist das Attribut auf „always“ gesetzt, wird die Ressource gestartet, egal, welchen Status sie beim Stoppen des Knotens hatte. Der Wert „restore“ bewirkt, dass die Clusterware den letzten bekannten Status wiederherstellt. „never“ legt fest, dass sie nie beim Serverstart gestartet wird
- *CHECK_INTERVAL*
  Das Sekundenintervall für die CHECK-Aufrufe
- *FAILOVER_INTERVAL sowie FAILOVER_THRESHOLD*
  Diese beiden geben an, ab wann eine Ressource als fehlerhaft angesehen wird. Wenn als Beispiel „FAILOVER_INTERVAL“ auf „60“ und „FAILOVER_THRESHOLD“ auf „5“ gesetzt wird, kann die Ressource

innerhalb von 60 Sekunden viermal als „fehlerhaft“ festgestellt werden, bevor beim fünften Mal automatisch ein Failover eingeleitet wird.
- *PLACEMENT*
  Ist das Attribut auf „balanced“ gesetzt, wird die Ressource auf einem beliebigen Server gestartet, der noch Kapazitäten frei hat. Wie hoch die Kapazität von Ressourcen ist, wird über das Attribut „LOAD“ angegeben. Wenn „favored“ gesetzt ist, werden als erstes die Server aus den Listen „SERVER_POOLS“ und „HOSTING_MEMBERS“ in Betracht gezogen. Ist keine der Maschinen möglich, wird sie auf einem beliebigen anderen Knoten gestartet. „restricted“ legt fest, dass sie nur auf einem Knoten einer der beiden oben genannten Listen gestartet werden darf. In diesem Fall darf nur „SERVER_POOLS“ oder „HOSTING_MEMBERS“ gesetzt sein. Ist kein Server aus der entsprechenden Liste vorhanden, wird die Ressource nicht gestartet
- *SCRIPT_TIMEOUT*
  Das Attribut legt fest, wie viele Sekunden ein Entry-Point laufen darf, bis „ABORT“ ausgeführt wird
- *START_DEPENDENCIES und STOP_DEPENDENCIES*
  Bei den Abhängigkeiten wurden sehr große Veränderungen eingeführt. In den älteren Versionen gab es nur das Attribut DEPENDENCIES. Jetzt wurden sowohl Start- als auch Stopp-Abhängigkeiten eingeführt
- *UPTIME_THRESHOLD*
  Gibt an, nach wie viel Zeit die Clusterware eine Ressource als stabil ansieht. Nach Ablauf wird der interne Zähler für Startversuche auf „0“ zurückgesetzt. Die Angabe wird durch eine Zahl, gefolgt von einem Buchstaben, spezifiziert. „s“ bedeutet Sekunden, „m“ Minuten, „h“ Stunden, „d“ Tage und „w“ Wochen

Ressource-Status

Der Status einer Ressource kann zu den bisherigen „ONLINE“, „OFFLINE“ und „UNKNOWN“ nun zusätzlich im Status „INTERMEDIATE“ sein. Dieser entsteht durch zwei Ereignisse:

- Die Clusterware ist nicht in der Lage, den Status einer Ressource festzustellen, sie war jedoch beim letzten CHECK im Begriff online zu gehen oder war bereits online
- Eine Ressource ist nur teilweise online. Dies kann zum Beispiel passieren, wenn eine Datenbank gestartet wird, sie allerdings noch im MOUNT-Status steht, beispielsweise wenn ein Roll-Forward durchgeführt wird

Abhängigkeiten

Wie bereits erwähnt, wurden die DEPENDENCIES in Start- und Stopp-Abhängigkeiten unterteilt. Damit wird der Online- und Offline-Vorgang sehr flexibel. Die Syntax ist jeweils gleich:

```
[START|STOP]_DEPENDENCIES=
dependency(resource_set) [de-
pendency
(resource_set)] […]
```

Jedoch sind unterschiedliche Werte für Start und Stopp erlaubt. Für die Start-Abhängigkeiten sind diese:

- *hard*
  Die Ressourcen oder Ressource-Typen, die in einem solchen „resource_set“ angegeben werden, müssen zwingend online sein, damit die Ressource gestartet werden kann
- *weak*
  Diese „dependency“ sorgt dafür, dass die im „resource_set“ angegebenen Ressourcen ebenfalls gestartet werden, falls sie OFFLINE sind. Ob diese Ressourcen jedoch erfolgreich gestartet wurden oder nicht, ist nicht von Belang. Es wird allerdings auf das Ergebnis des ONLINE-Befehls gewartet
- *attraction*
  Wird eine solche Abhängigkeit angegeben, so wird versucht, die Ressource auf dem gleichen Server zu starten wie die im „resource_set“ angegebenen Ressourcen oder Ressource-Typen
- *pullup*
  In diesem Fall wird die Bedeutung von „weak“ umgekehrt. Eine Ressource, die eine „pullup“-Dependency angibt, wird automatisch gestartet, wenn die im „resource_set“ angegebenen Ressourcen gestartet werden und sie selbst noch „OFFLINE“ ist
- *dispersion*
  Diese Abhängigkeit ist die Umkehrung von „attraction“. Eine Ressource mit dieser Abhängigkeit wird nach Möglichkeit NICHT auf einem Knoten gestartet, auf dem eine im „resource_set“ angegebene Ressource online ist

Bei der Angabe von Abhängigkeiten lassen sich alle oben angegebenen Varianten beliebig kombinieren. Außerdem kann man verschiedene Modifikatoren angeben, die das Verhalten ändern, siehe [3]. Für die Stopp-Abhängigkeiten ist hier nur der Wert „hard“ erlaubt, der angibt, dass alle abhängigen Ressourcen gemeinsam gestoppt werden müssen.

Beispiel Tomcat-Server

In diesem Beispiel wird ein Tomcat-Server unter Cluster-Kontrolle gebracht. Dabei werden folgende Abhängigkeiten eingehalten:

- Es wird eine virtuelle IP-Adresse für den Zugriff verwendet. Diese wird als Erstes gestartet. Wenn sie nicht mehr zur Verfügung steht, werden alle abhängigen Ressourcen automatisch gestoppt
- Der Tomcat-Server ist auf einem Filesystem installiert, das auf einer SAN-Platte liegt, die an allen Knoten sichtbar ist. Das darauf liegende Dateisystem darf aber immer nur auf einer Maschine gemountet werden
- Sind Filesystem und IP-Adresse vorhanden, wird der Tomcat-Dienst zur Verfügung gestellt

Um die VIP hinzuzufügen wird folgender Befehl verwendet (siehe Listing 1).

Das notwendige Attribut „USR_ORA_VIP“ gibt die IP an, die vom Typ „ora.cluster_vip_net1.type“ ist. Die Ressource ist abhängig von „ora.net1.network“ und kann (hard) nur online gehen, wenn diese Ressource bereits online ist und wird (pullup) automa-

```
crsctl add resource tomcat_vip -type ora.cluster_vip_net1.type -attr „USR_ORA_VIP=192.160.2.96, START_
DEPENDENCIES=‘hard(ora.net1.network) pullup(ora.net1.network)‘, STOP_DEPENDENCIES=‘hard(ora.net1.network)‘
```

Listing 1

```
crsctl add type fs_type -basetype cluster_resource \
-attr „ATTRIBUTE=FS_DEVICE,TYPE=string,FLAGS=REQUIRED“ \
-attr „ATTRIBUTE=FS_MOUNT_POINT,TYPE=string,FLAGS=REQUIRED“ \
-attr „ATTRIBUTE=FS_MOUNT_OPTIONS,TYPE=string“ \
-attr „ATTRIBUTE=FS_TYPE,TYPE=string,FLAGS=REQUIRED“ \
-attr „ATTRIBUTE=ACTION_SCRIPT,TYPE=string, DEFAULT_VALUE=/root/fs.sh,FLAGS=READONLY“
```

Listing 2

```
crsctl add resource TC_FS -type fs_type -attr „FS_DEVICE=/dev/sdaa,FS_MOUNT_POINT=/opt/TC,FS_
TYPE=ext3,FS_MOUNT_OPTIONS=‘‘,START_DEPENDENCIES=‘hard(tomcat_vip) pullup(tomcat_vip)‘,STOP_
DEPENDENCIES=‘hard(tomcat_vip)‘“
```

Listing 3

```
crsctl add type tc_type -basetype cluster_resource \
-attr „ATTRIBUTE=TOMCAT_USER,TYPE=string,FLAGS=REQUIRED“ \
-attr „ATTRIBUTE=START_SCRIPT,TYPE=string,FLAGS=REQUIRED“ \
-attr „ATTRIBUTE=STOP_SCRIPT,TYPE=string,FLAGS=REQUIRED“ \
-attr „ATTRIBUTE=ACTION_SCRIPT,TYPE=string, DEFAULT_VALUE=/root/tomcat.sh,FLAGS=READONLY“
```

Listing 4

```
crsctl add resource tomcat -type tc_type -attr „TOMCAT_USER=tc,START_SCRIPT=/root/tc_startup.sh, STOP_
SCRIPT=/root/tc_shutdown.sh,START_DEPENDENCIES=‘hard(TC_FS) pullup(TC_FS)‘, STOP_DEPENDENCIES=‘hard(TC_
FS)‘“
```

Listing 5

tisch gestartet, wenn ora.net1.network gestartet wurde. Die Ressource „ora.net1.network“ wird bei der Installation automatisch angelegt und stellt das öffentliche Netzwerkinterface dar. Anschließend wird ein neuer Typ für das Filesystem angelegt (siehe Listing 2).

Dieser Typ basiert auf „cluster_resource“, und die Attribute „FS_DEVICE“, „FS_MOUNT_POINT“, „FS_MOUNT_OPTIONS“ und „FS_TYPE“ müssen beim Erstellen angegeben werden. Das Action-Skript wird hierbei fest eingestellt (FLAGS=READONLY). Danach kann ein Filesystem angelegt werden (siehe Listing 3).

Die Ressource basiert auf dem angelegten Typ; das Filesystem auf dem Device „/dev/sdaa“ vom Typ „ext3“ wird beim Starten nach „/opt/TC“ ohne Optionen gemountet. Gestartet wird es automatisch, sobald „tomcat_vip“ online gegangen ist. Es wird gestoppt beziehungsweise restartet, wenn die IP offline geht. Als nächstes wird ein Typ für den Server angelegt:(siehe Listing 4).

Der Typ basiert wieder auf der „cluster_resource“ und hat drei benötigte Attribute, über die der Benutzer angegeben wird, unter dem der Prozess läuft, sowie zwei externe Skripte, die zum eigentlichen Starten und Stoppen genutzt werden. Als letztes kann dann der Tomcat-Server angelegt werden (siehe Listing 5).

Hierbei werden der Benutzer auf „tc“ festgelegt sowie die beiden Skripte angegeben. Die Ressource kann wieder nur starten, wenn das Filesystem gemountet ist und wird dann auch sofort online gesetzt. Wenn das Filesystem offline geht, wird auch der Tomcat gestoppt.

Referenzen

[1] My Oracle Support: Note 1053147.1

[2] Using Oracle Clusterware to Protect A Single Instance Oracle Database 11g: http://www.oraclecom/technetwork/database/si-db-failover-11g-134623.pdf

[3] Oracle Clusterware Administration and Deployment Guide 11g Release 2 (11.2): Oracle Clusterware Resource Reference unter http://download.oracle.com/docs/cd/E11882_01/rac.112/e16794/resatt.htm#CHDCECHH

Kontakt:
Jörg Horchler
joerg.horchler@bauermedia.com