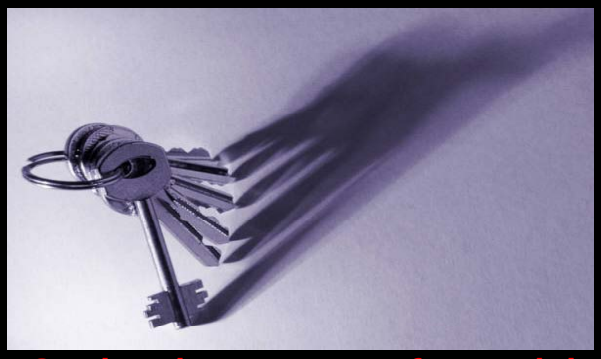
Oracle10g RAC의 CRS Overview



Getting the most out of MetaLink

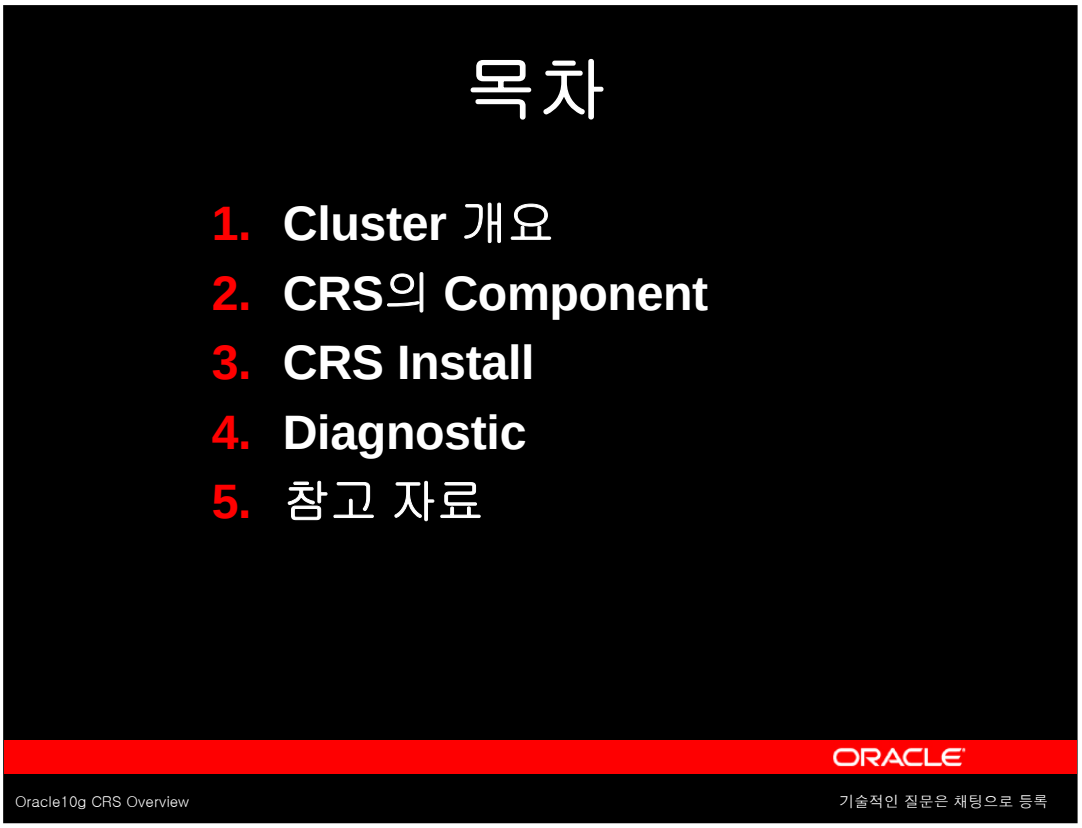
남궁혁

한국오라클 (주) 제품지원실

ORACLE

Oracle10g CRS Overview 기술적인 질문은 채팅으로 등록

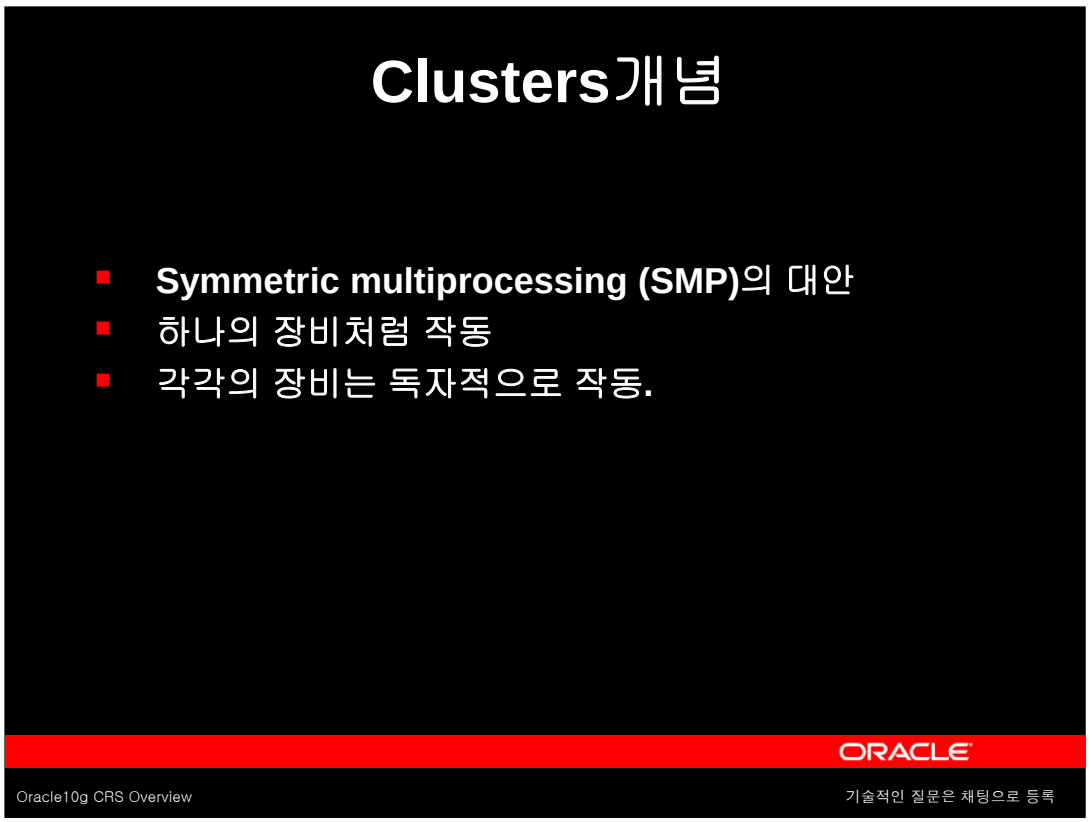
Oracle 10g version부터 RAC에서 사용하는 cluster manager를 Oracle에서도 제공을 해주는데, 이 Oracle cluster manager가 CRS입니다. 이렇게 cluster manager를 CRS를 사용하게 되어 서로 다른 platform에 대해서 동일한 interface를 갖을 수 있습니다.



CRS의 내용 자체가 상당히 많기 때문에 오늘 진행하는 iSeminar에서는 많은 내용을 볼 수는 없고, 간략하게 CRS의 구성 component들이 무엇이 있는지, 그리고 각 component들의 역할이 무엇인지에 대해 알아보겠습니다.

마지막 부분에는 CRS에서 문제가 의심될 때 어떤 trace file을 보아야 하는지에 대해서도 확인해보려고 합니다.

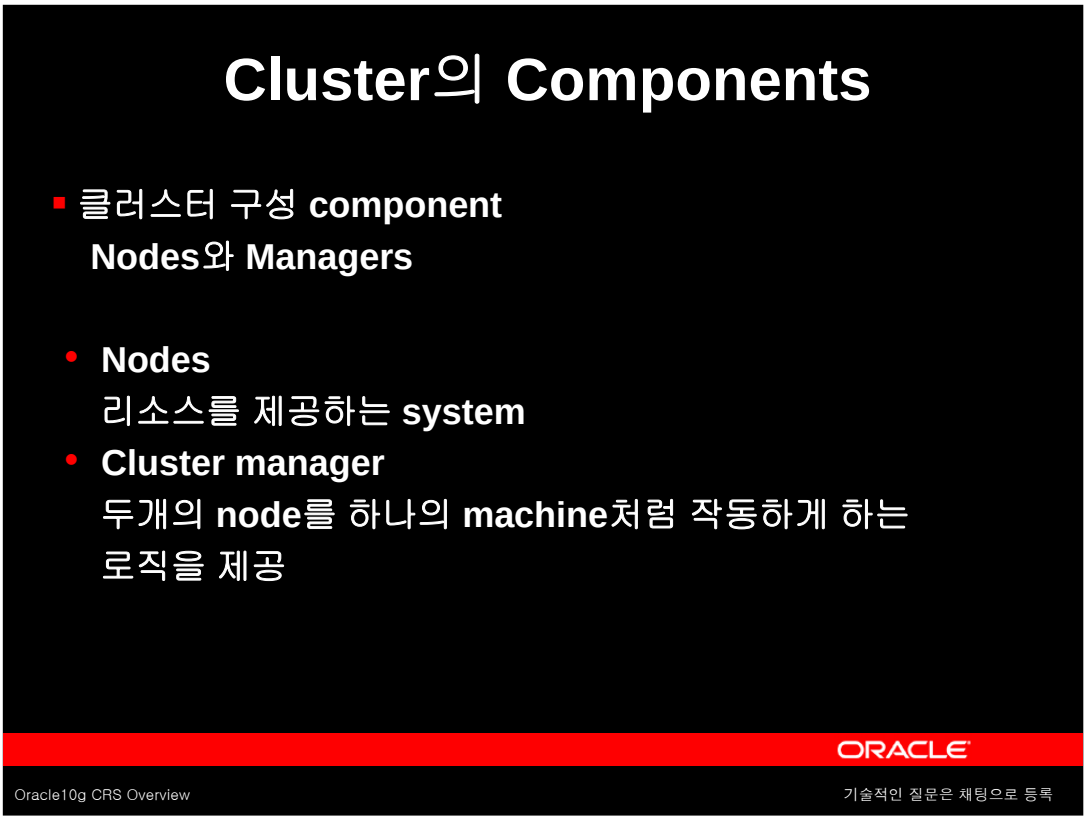
세미나는 약 25분 소요될 것으로 예상되며 질문은 발표 마친 후 별도로 받도록 하겠습니다.
그럼 지금부터 세미나를 시작하겠습니다.



클러스터란, 대칭형 다중 처리 방식(symmetric multiprocessing 또는 SMP)의 대안으로, 상호 연결된 하나 이상의 컴퓨터가 그룹을 이루어 작업을 함께 처리하는 방식입니다.

시스템을 이용하는 사용자 입장에서는 하나의 시스템을 사용하는 것과 같은 효과를 나타냅니다.

즉 최종 사용자 입장에서는 사용중인 시스템이 물리적으로 한대의 노드로 구성되어 있는지, 또는 여러 대의 노드가 클러스터를 구성하여 서비스를 제공하는지 고려하지 않아도 되는 것입니다.



클러스터는 두개의 주요 요소로 구성됩니다.

클러스터 노드는 작업을 처리하는데 필요한 자원을 제공해 주는 시스템입니다.

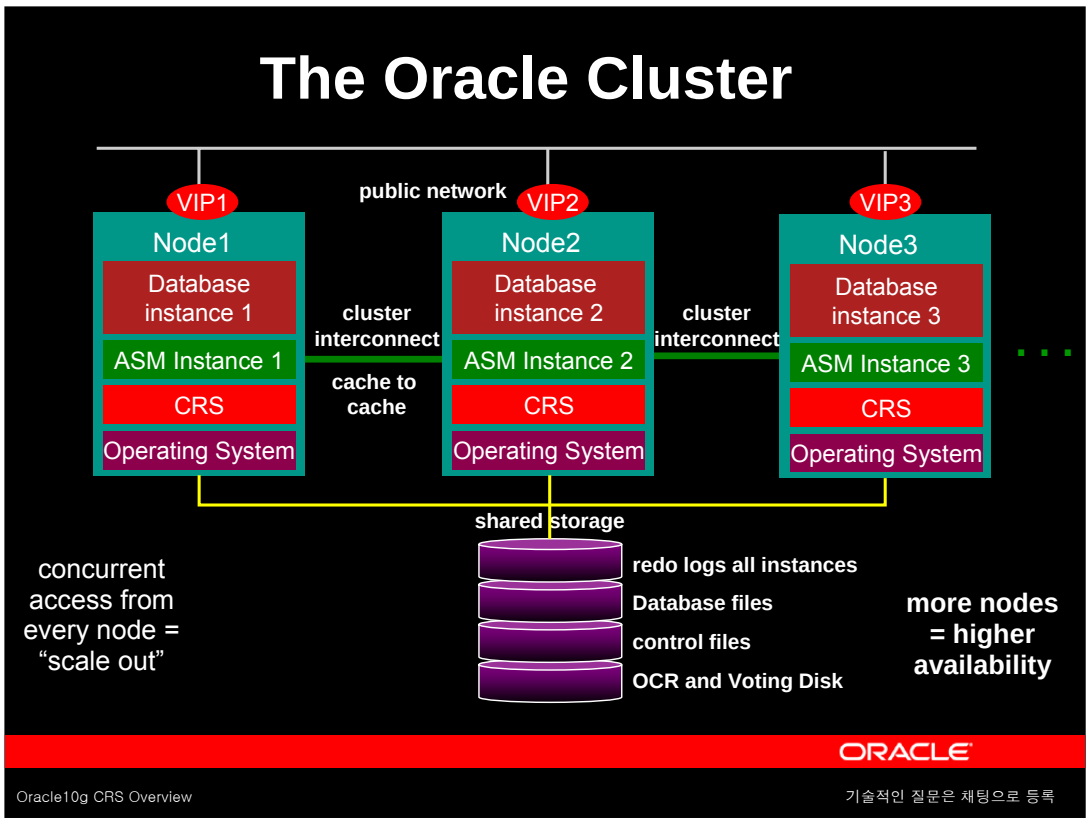
즉, application이 작업할 수 있는 하드웨어를 말합니다. 이 하나이상의 노드를 논리적으로 하나의 시스템으로 묶어주는 software가 필요하게 됩니다.

클러스터 매니저가 이 역할을 해 주는 시스템 소프트웨어입니다.

일반적으로, 클러스터를 구성하는 노드 관리, 클러스터 내 노드 추가, 삭제 , 자원 모니터링, Failover 처리 기능을 갖습니다.

Cluster내에서 작동하는 software는 cluster system에서 작동할 수 있도록 설계되어야 합니다.

즉, stand alone node에서 작동하도록 설계된 application은 cluster system에서 작동하지는 않습니다.



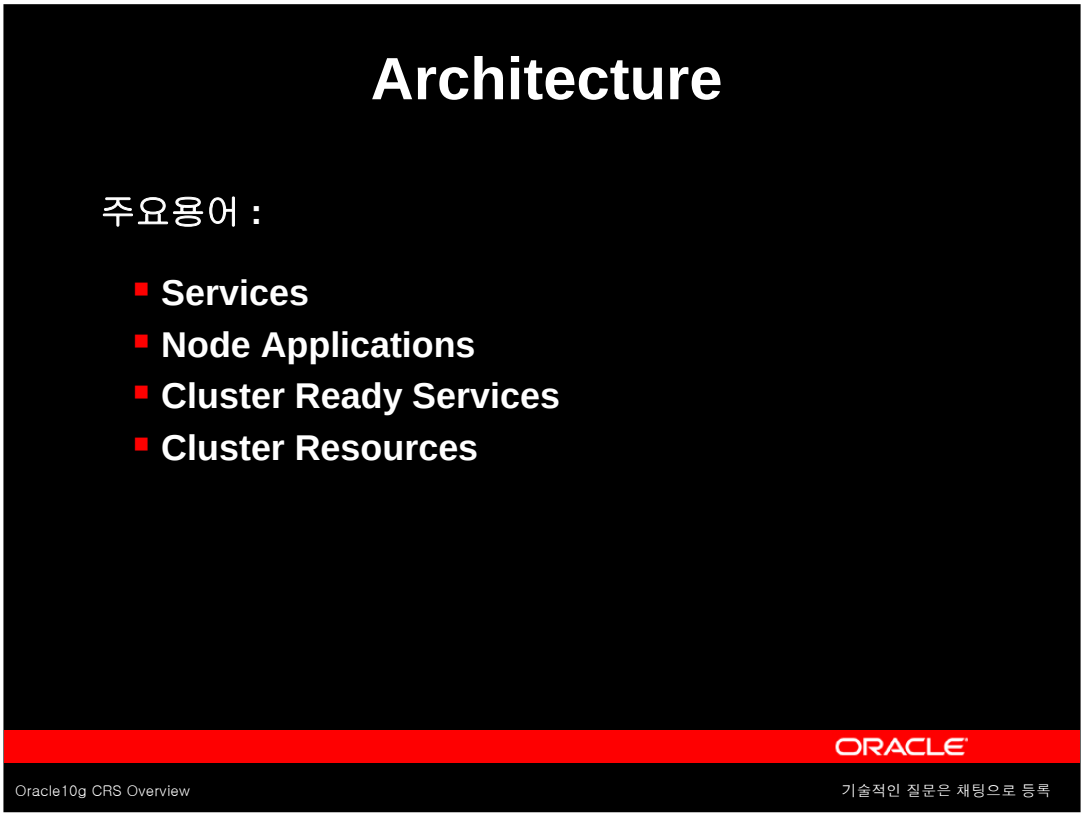
여기 간단한 그림으로 우리는 Oracle cluster의 콤포넌트들을 볼 수 있습니다.

OCR과 Voting disk도 또한 oracle cluster의 콤포넌트입니다. 스토리지에서 제일 하단의 ocr, voting disk부터 controlfile, database files의 순서로 oracle은 file을 check하면서 read하기 시작합니다.

이때 storage는 cluster내의 모든 node에서 동시에 access가능해야 합니다.

여기서 CRS가 전체 node에 걸쳐 작동하고 있는 것을 알 수 있습니다. ASM으로 구성된 경우라면 ASM instance가 필요하게 되고, 이 모든 Layer들이 정상적으로 start된 이후 oracle database instance가 작동하게 됩니다.

Public network망에서 interface는 일반 public IP를 사용하지 않고, VIP를 사용하게 됩니다.
이것은 IP failover를 위해 ORACLE CRS가 제공하는 것 입니다.



오라클 클러스터 아키텍처를 이해하는데 필요한 중요한 개념들이 있습니다.

서비스, Node Application, Cluster Ready Services, Cluster Resources등입니다.

먼저 Service는 클러스터 내 작업을 배분해 주는 수단을 제공해 줍니다.

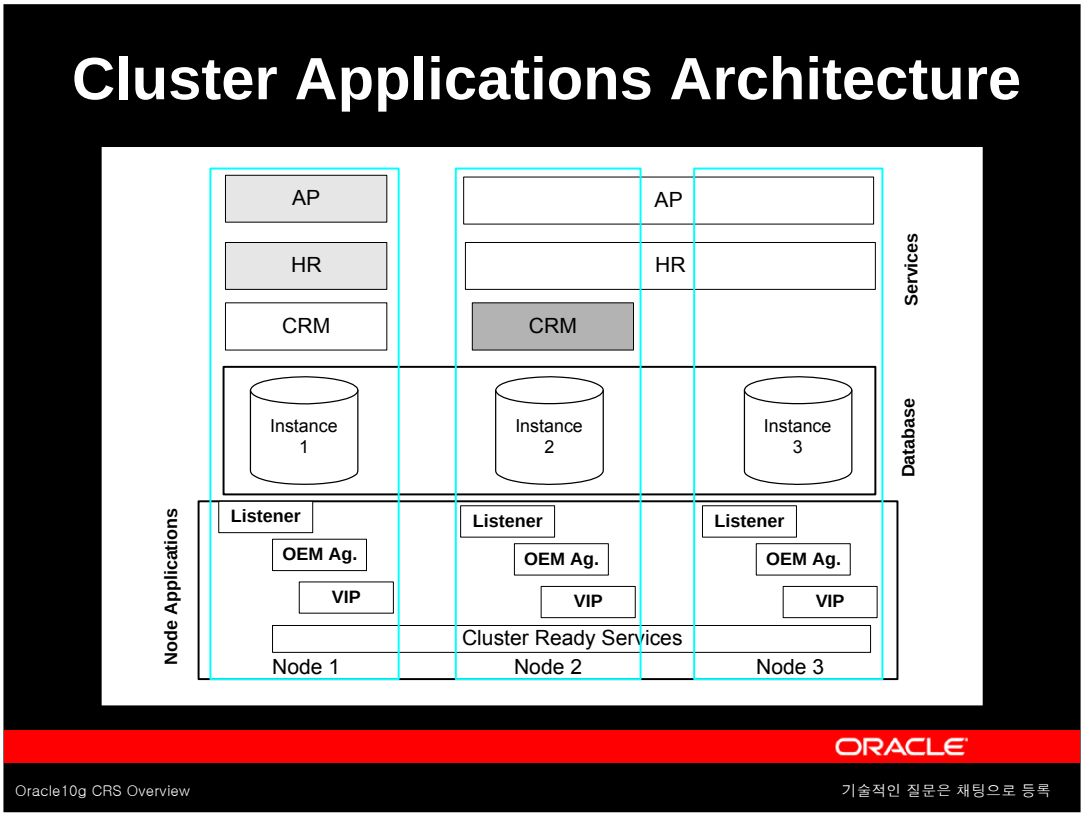
서비스는 하나 또는 그 이상의 인스턴스에서 제공되는데, 데이터베이스 명이 기본적으로 하나의 서비스입니다.

Oracle 10g이전 버전에서도 이 Service라는 개념이 있었지만, 10g에서는 더 확대된 개념으로 하나의 instance에서 여러 Service를 제공할 수 있고, 하나의 서비스가 여러 instance에 걸쳐 제공될 수도 있습니다.

Node Application은 Oracle Listener, Global Services Daemon (GSD), Oracle Enterprise Manager Agent (OEM Agent), Virtual IP Address (VIP) 등이 포함됩니다.

Cluster Ready Service는 logical하게 node를 묶어주는 Cluster Manager이고, Cluster Resource는 Oracle Cluster Service에 의해 관리되는 Resource들이며 데이터베이스 인스턴스, 서비스, Listener 및 VIP 등이 포함됩니다.

Resource의 속성은 failover 특성 및, 재 구동 방식, 의존 관계 등을 지정 하기 위해 정의 할 수 있습니다.



이 그림은 앞에서 설명한 몇 가지 용어를 그림으로 표시한 것 입니다.

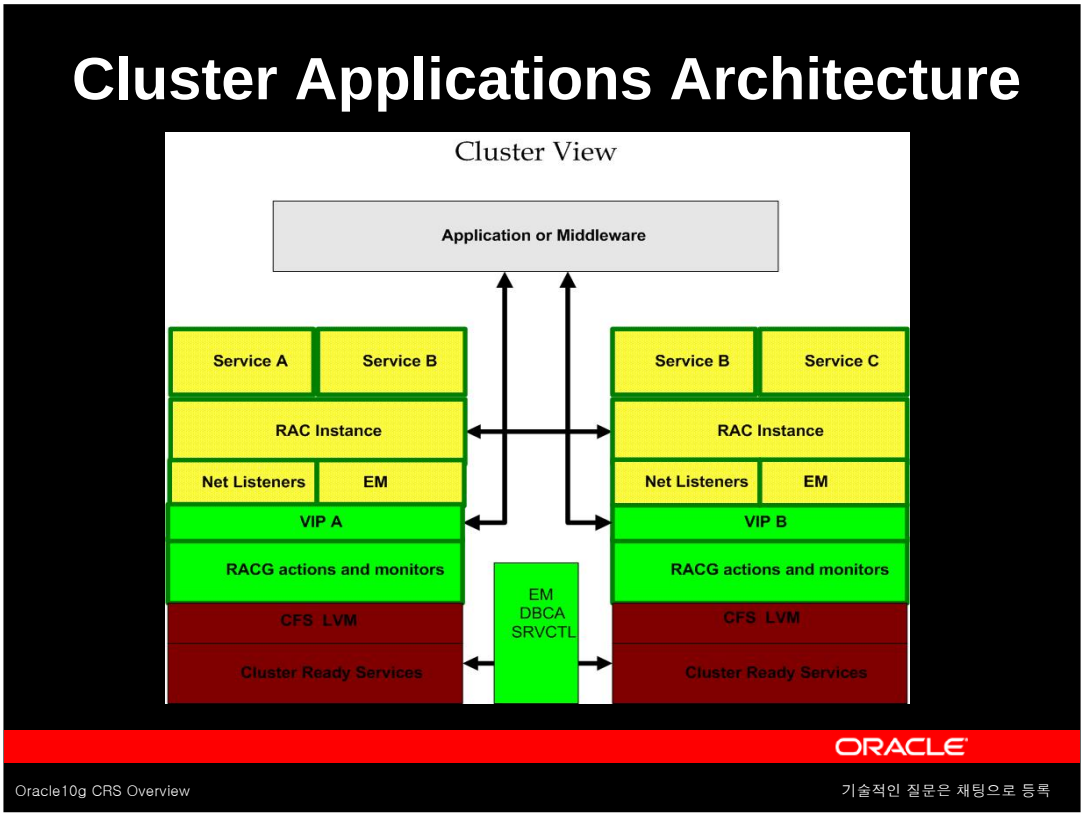
3개의 node를 사용하고있고, 각 node들은 기본적으로 Cluster Ready Services에 의해 하나의 system으로 clustering되어있습니다.

각 Node application들이 각각의 node에서 작동하고있고, Database를 위한 Instance가 각 node에 올라와 있습니다.

Service를 보면, 이 Database는 AP, HR, CRM 3개의 Service가 있습니다.

회색으로 표시된 것은 failover시 작동할 Service입니다. CRM을 예를 들면 평시에는 Node1에서 Service를 제공하고, 만약 Node1에 이상이 있을 경우 Node2에서 CRM service를 제공하게 됩니다.

이후 Fail된 Node가 복구되면 다시 Service는 'preferred' Node로 돌아올 수 있습니다. 단, 이 경우는 user가 manual하게 옮겨주어야 합니다.



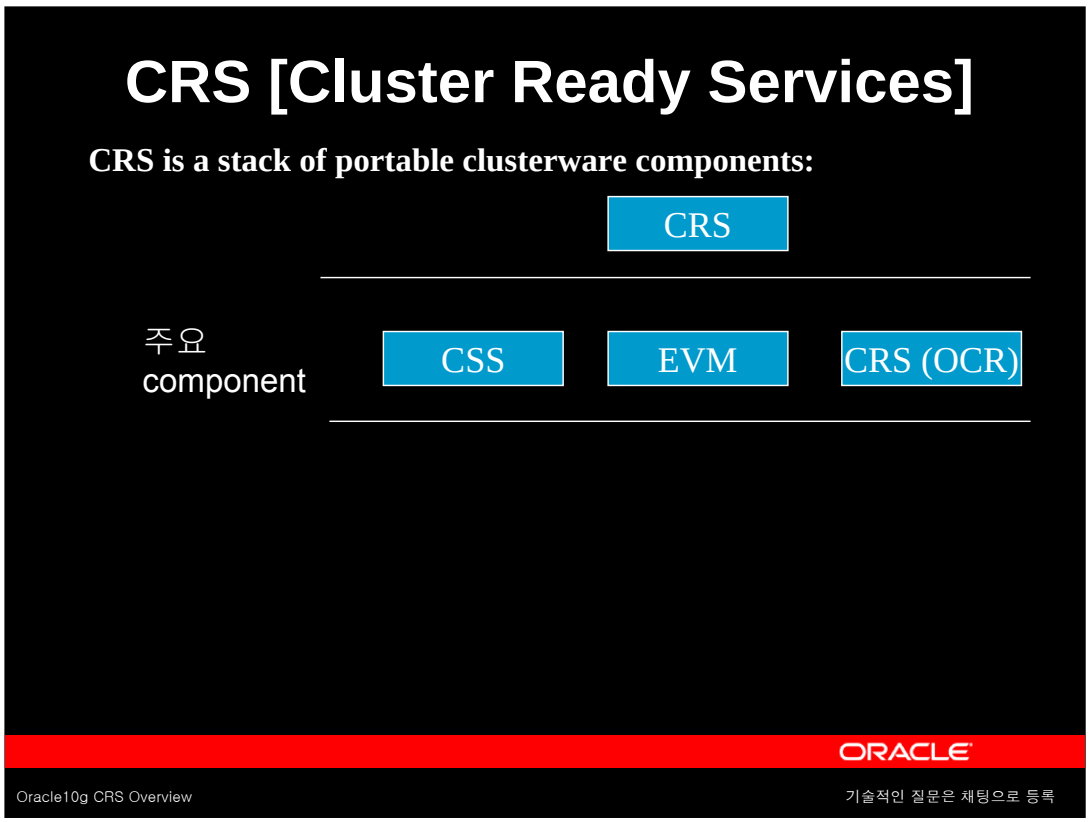
앞의 그림은 Service의 관점에서 그린 그림이고, 이 그림은 CRS와 Oracle Instance, Application의 관계를 보여줍니다.

Oracle Database Instance가 각 node에 하나씩 작동하고 있고, 이 두개의 Instance가 Service A, B, C 3개의 Service를 제공합니다. 이 그림에서는 두개의 Instance가 모두 작동하고 있지만, 어떤 경우는 다른 하나의 Instance는 Standby용 혹은 Failover용으로 대기중일 수도 있습니다.

그렇다고 하더라도 Node Application, 즉, Listener, EM, GSD등은 여전히 각 node별로 작동하고 있습니다.

Client Application은 기본적으로 VIP를 통하여 접속이 되고, Oracle Listener도 이 VIP를 listen하게 됩니다.

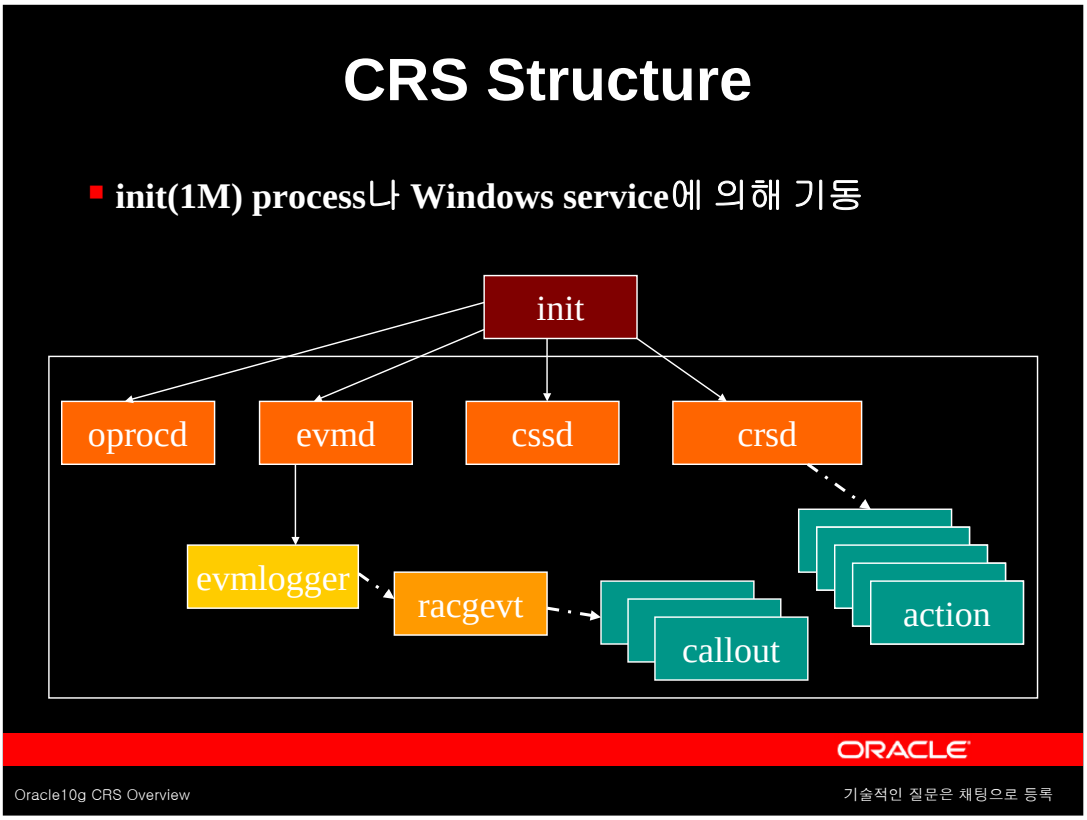
EM, DBCA, SRVCTL, NETCA등은 HA적 관점에서 CRS와 통신하게 되며, 새로운 Resource들을 생성할 때 CRS와 통신하여 OCR에 정보를 기록하게 합니다. 이 OCR과 voting Disk는 OS storage에 위치하게 됩니다.



이제 본격적으로 CRS의 구성 콤포넌트들에 대해 알아보겠습니다.

CRS는 크게 3개의 콤포넌트로 구성이 됩니다.

CSS, EVM, CRS. 이렇게 3개가 존재하며, 마지막의 CRS는 Oracle Cluster를 부를 때 사용하는 CRS와는 구분이 되는 Process명으로써, CRS의 resource를 관리합니다.



이 그림은 앞의 그림들 좀더 확장한 그림입니다. CRS는 Unix의 init에 의해 기동 되고, windows에서는 service controller에 의해 기동 됩니다.

여기서 하얀 box내의 사각형들은 모두 CRS의 component들입니다.

Oprocd는 oracle10g에서 사용하는 io fencing기능을 제공합니다.

Evmd는 cluster내의 event발생시 이 event를 전파하는 기능을 합니다.

이 그림에 표시되어 있지는 않지만, 'crs_stat –t'명령을 수행하면 gsd(group service daemon)가 나타나는 것을 보실 수 있습니다. Gsd는 10g version에서는 사용되지 않지만, 9i version의 srvctl의 interface가 gsd로 되어있기 때문에 10g에서도 아직 남아 있습니다.

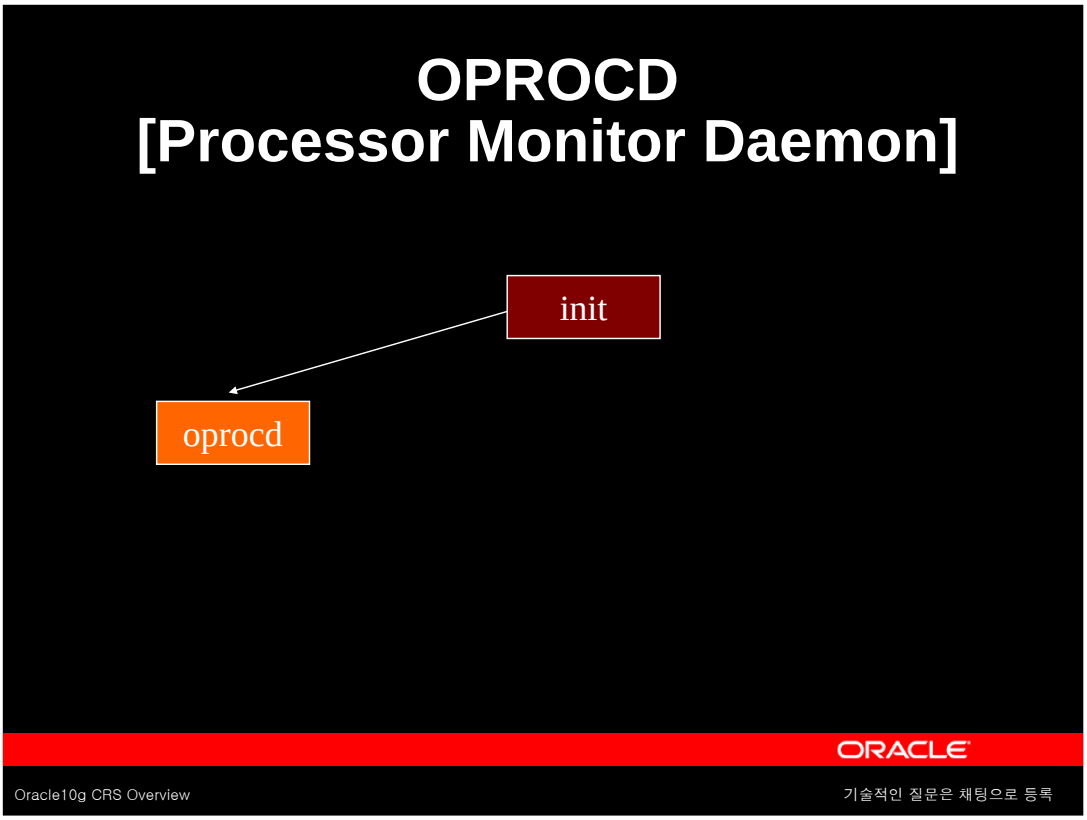
실제로 oracle9i때의 gsd의 역할은 이제 모두 crsd가 담당하게 되었고, 9i version을 지원하기 위해 존재하는 것 이외의 의미는 없습니다.

Ocssd는 heartbeat을 다른 node에 보내어 health check를 하는 기능을 담당하며, 상대 node의 이상 감지시 cluster reconfiguration도 css가 담당합니다.

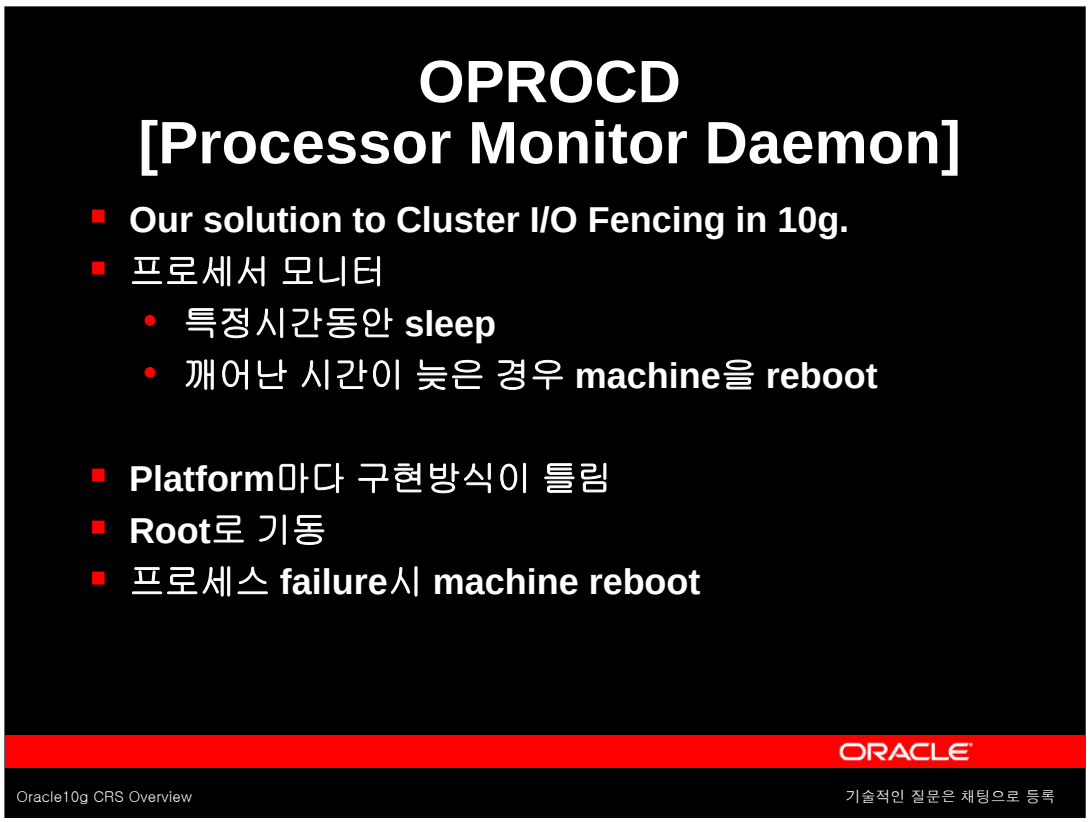
Crsd는 cluster resource manager로써 CRS가 사용하는 resource를 start/stop/disable/configure하는 역할을 합니다.

Process로서 존재하지는 않지만, OCR과 Voting disk도 CRS의 구성 Component입니다.

OCR은 cluster와 cluster내의 resource의 정보가 저장되어있으며, Voting Disk는 각 node의 status를 확인하기위해 사용됩니다.



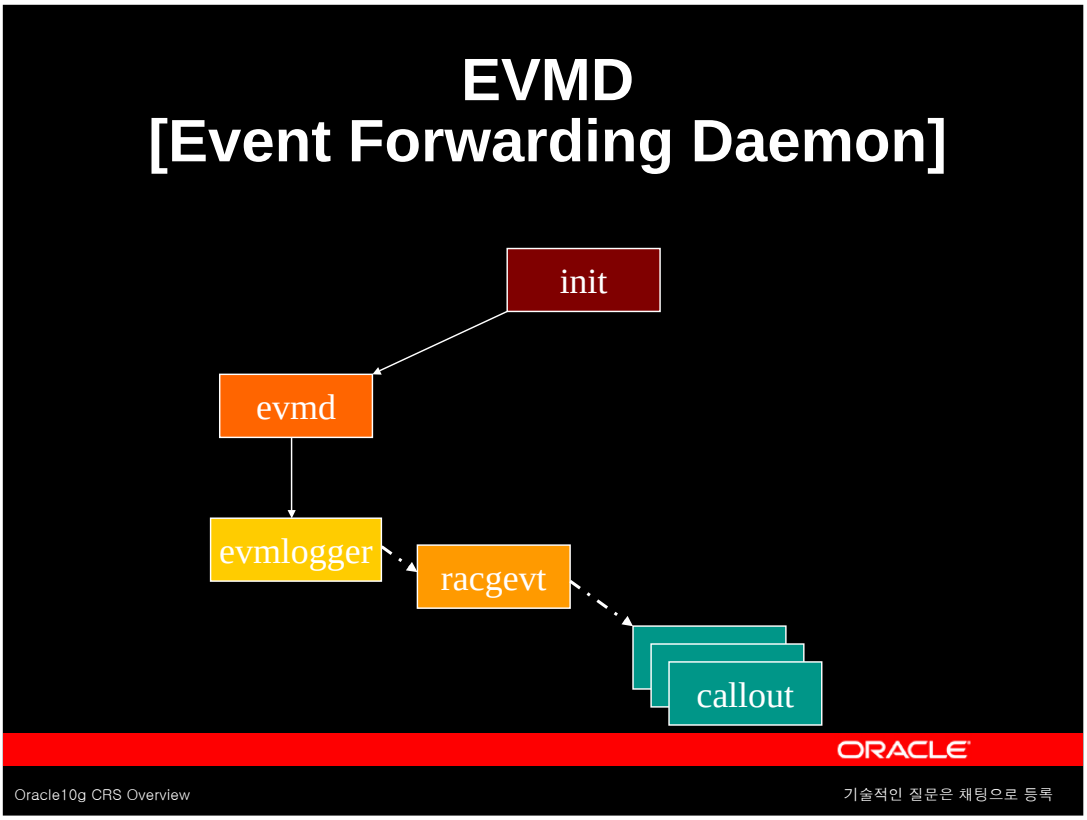
Oprocd는 processor monitor로써 init에 의해 기동됩니다.



앞에서 말한 것처럼 oprocd는 Oralce Cluster가 제공하는 IO fencing solution입니다.

Linux에서는 hangcheck timer가 이 oprocd의 기능을 제공하기때문에 oprocd는 작동하지 않습니다. 이외의 platform중에서 vendor clusterware가 설치되어 있다면, 마찬가지로 oprocd는 기동 되지 않습니다. Linux를 제외한 flatform에서 Vendor clusterware가 설치되지 않은 경우만 oprocd가 기동 되어 processor를 monitoring하게 됩니다.

Monitoring하는 방법은 특정시간동안 sleep후 깨어나서 예정된 시간보다 늦게 깨어나게 되면, system hang으로 간주하여 processor를 reset하고 machine을 reboot하게 됩니다.



EVMD는 Event forwarding Daemon입니다. 이 process도 init에 의해 기동됩니다.

Evmd는 evmlogger를 fork하는데 이 process는 log file에 event를 기록하는 역할을 합니다.

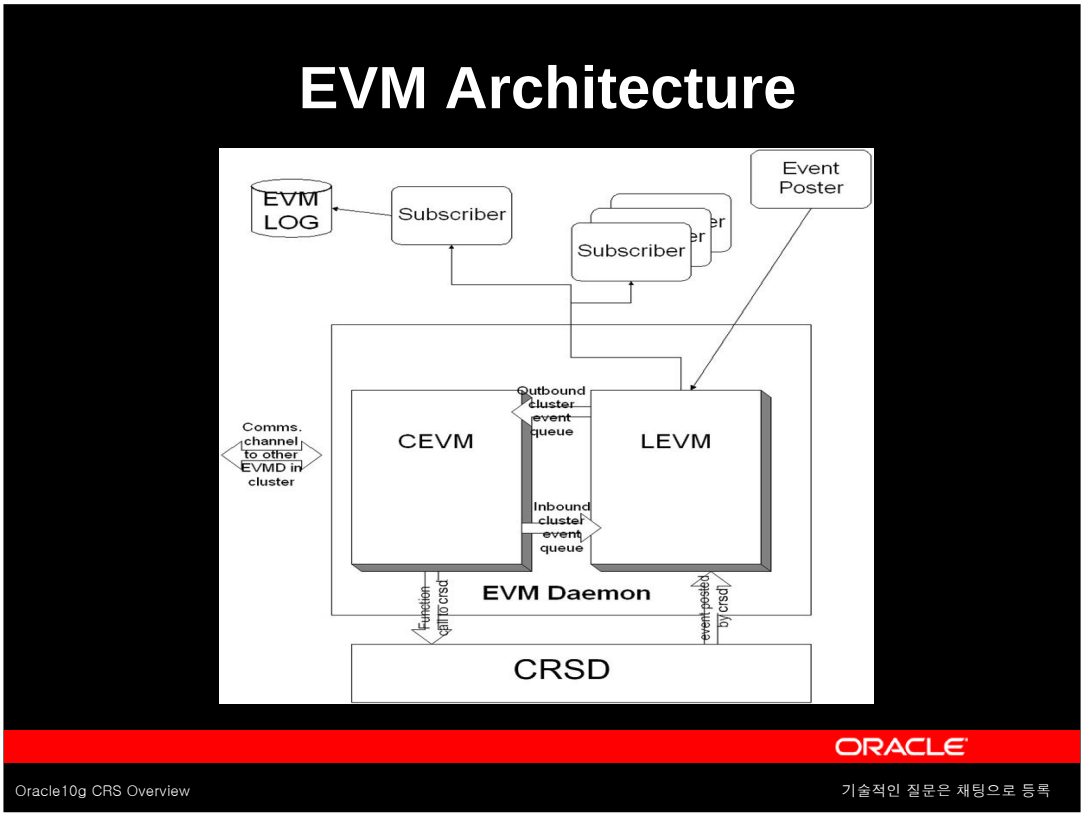
Oralce 10g에서는 oracle instance를 monitoring하는 racgimon이라는 process가 있는데, 이 process로부터 forwarding되는 message를 처리하기위해 evmlogger는 필요에 따라 racgevt라는 process를 생성합니다.

Racgevt는 message를 받을 때 마다 callout directory를 확인하여 필요 시 callout를 실행합니다.

Callout이란 특정 event에 따라 행해져야 하는 특정 action을 정한 script입니다.

Callout directory는 $ORA_CRS_HOME/racg/usrco입니다. 이 directory에 user가 특정 action을 미리 define하여 callout script를 생성할 수도 있습니다.

Evmd는 oracle user로 기동되며 fail발생시 자동적으로 재기동 됩니다.



EVM Daemon은 다시 두개의 component로 구분할 수 있습니다.

하나는 local EVM(LEVM)이고 하나는 cluster EVM(CEVM)입니다.

CEVM은 clusters내의 evm daemon에 대한 cluster membership정보를 유지관리 합니다.

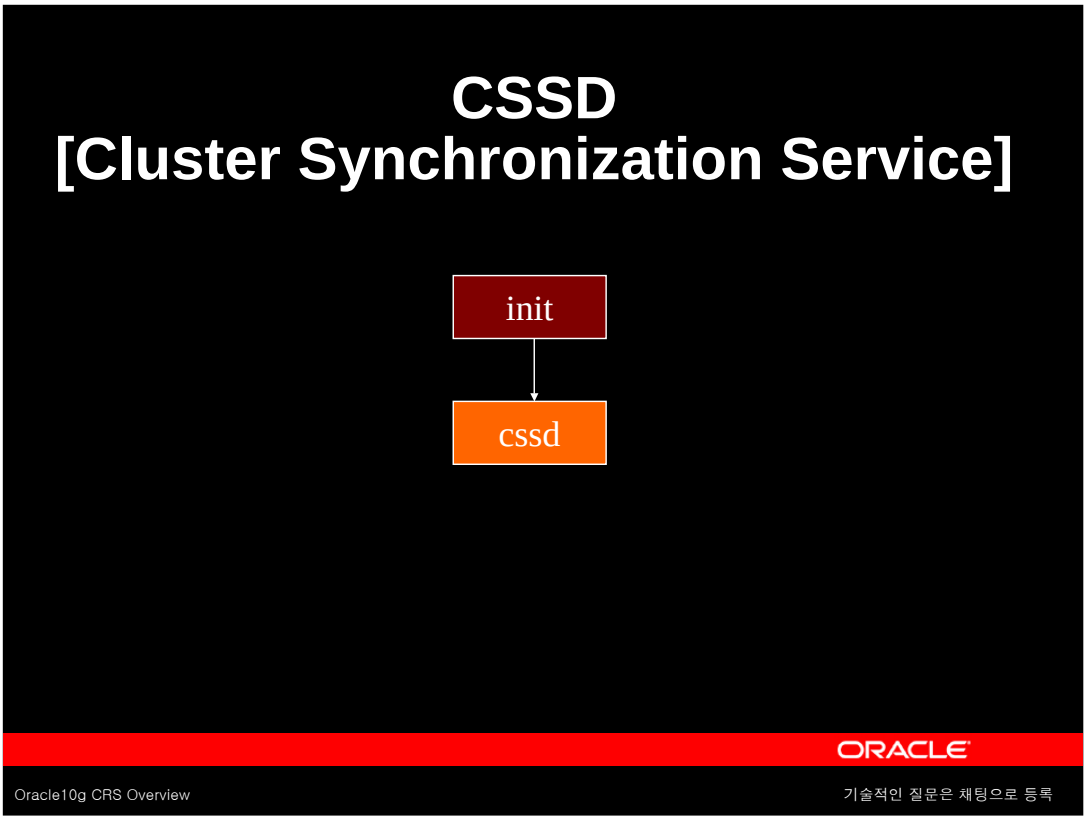
즉, 상태 node의 CEVM의 TCP connection정보를 유지하고, 상대 node의 evm에게 event를 전달하는 역할을 합니다.

LEVM은 clusterwide하게 전달될 event가 생성되면, Cluster event Queue에 event를 넣어서 CEVM이 clusterwide하게 event를 전달하도록 합니다.

CEVM이 다른 node로부터 event를 받는 경우, 마찬가지로 clustger event Queue에 event를 넣어 LEVM으로 전달합니다.

Cluster내의 어느 node가 새로 cluster에 join하는 경우, CEVM은 상대 node의 CEVM으로부터 이 event를 전달받고, inbound cluster event queue에 이 event를 넣습니다. 그리고 CRSD에게 이 event를 넘겨줘서 CRSD가 node join을 알게 합니다. LEVM은 queue에 있는 event를 받고 이 내용을 기록합니다.

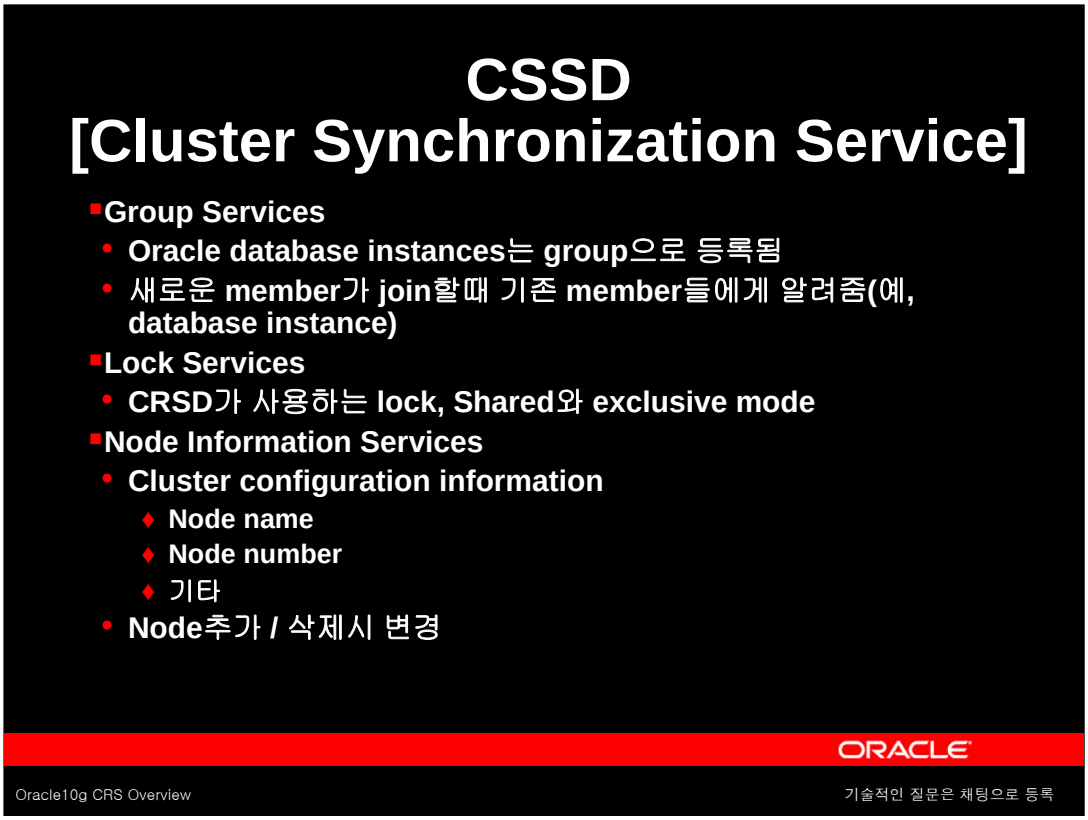
이와 같이 CSS, CRSD가 발행하는 event들은 EVM을 통하여 clusterwide하게 전파됩니다.



CSSD는 cluster synchronization service daemon의 약자입니다. 이 process는 다음 세가지의 service를 제공합니다.

Group service, lock service, node information service입니다. Node information service는 어떤 node가 cluster에 join했는지 혹은 cluster를 떠났는지 monitoring하는 기능입니다.

Cssd는 process fail시 OS reboot이 됩니다.

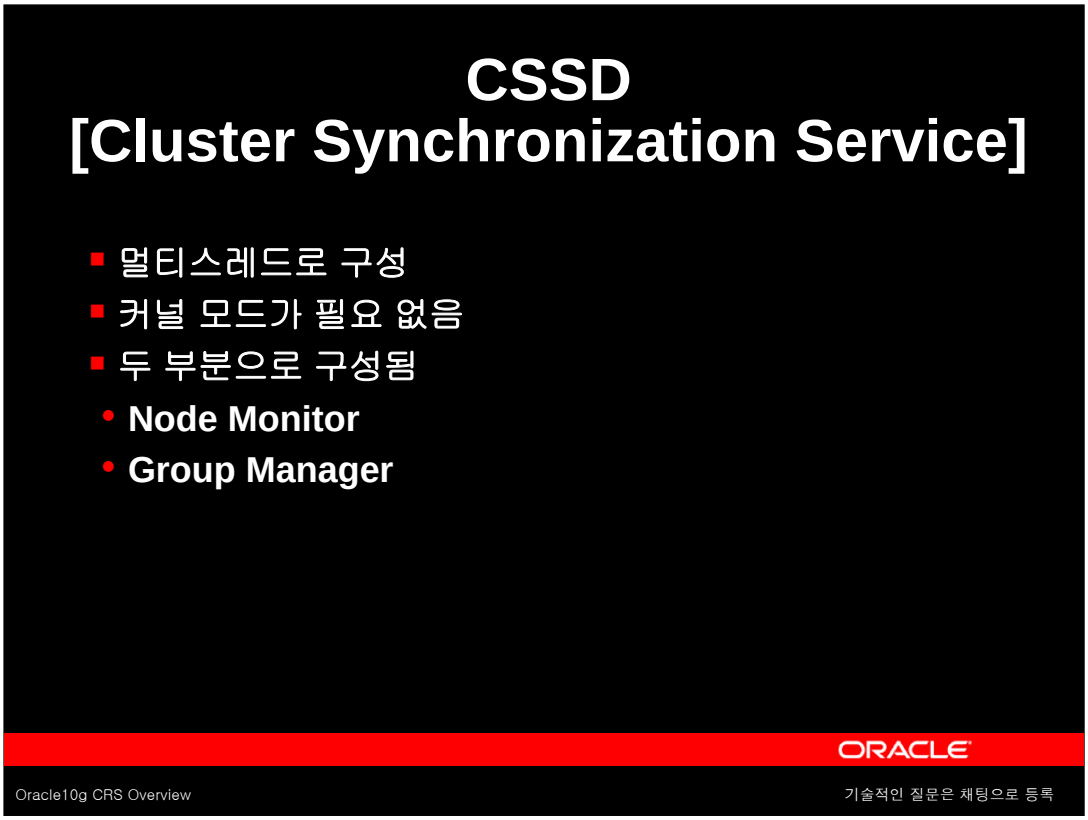


CSS의 역할인 그룹서비스는 9i version에서의 skgxn의 역할이 확장된 것입니다. Oracle database instance가 join하는 것을 처리하고 Cluster내의 이미 존재하는 member들에게 새로운 member가 join되거나 leave되는 것을 알려줍니다.

Lock service는 crsd가 사용하는 lock을 css의 lock service가 제공합니다.

Node Information Services는 node의 정보를 유지 관리하는 service입니다.

여기에는 cluster configuration정보, node명, node number등이 있으며 node추가나 삭제 시 갱신됩니다.



다른 crs process들처럼 cssd도 multi thread로 되어있으며, 크게 두 부분으로 나눌 수 있습니다.

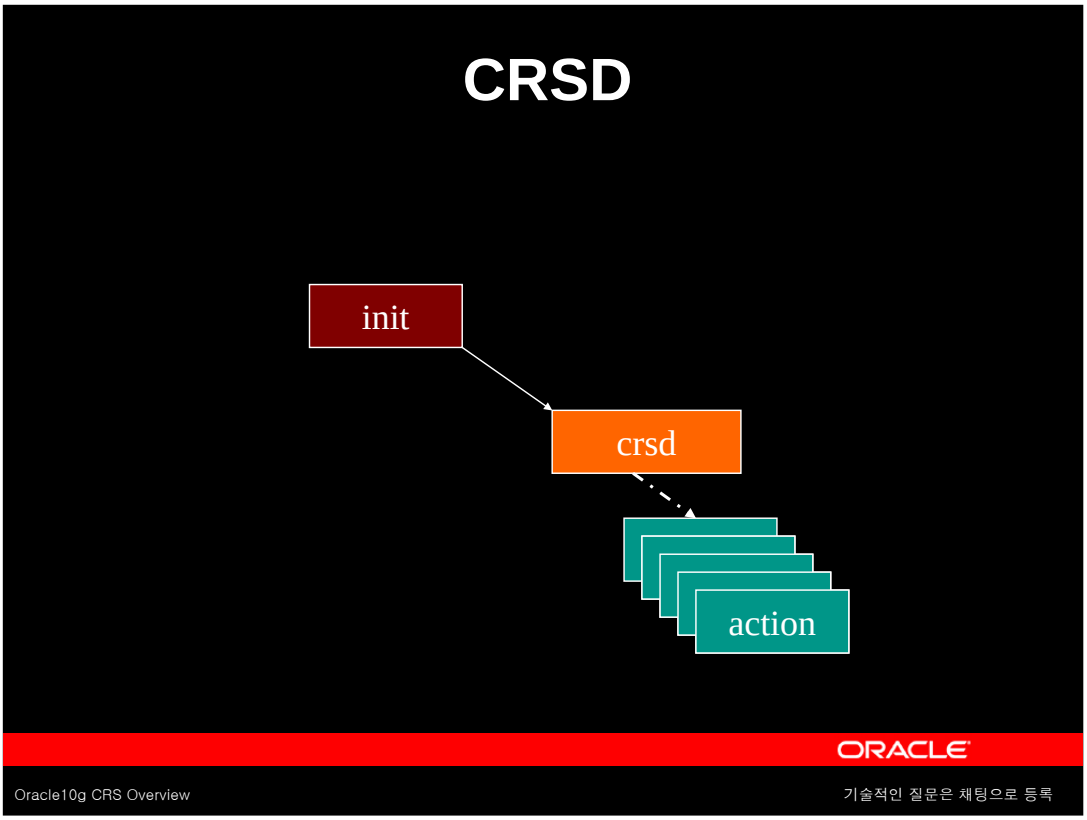
Node monitor는 nm이라고 불리우며, Node자체가 살아있는지 여부를 check한는 역할을 합니다. 이 기능은 heartbeat을 이용하게 됩니다.

상대 node의 css endpoint는 ocr에 기록되어있으며, 이 end point로 매초마다 interconnect를 통해 network heartbeat을 보냅니다. 일정 시간동안, 즉 miscount동안 heartbeat을 받지 못하면 css는 해당 node를 죽은 것으로 간주합니다. 이와는 별도로 voing disk을 매초마다 read하게 됩니다.

Network heartbeat이 miscount동안 오지 못하여 일부 node가 죽은 것으로 판단되면, 이 voting disk를 확인하여 어느 node들을 cluster에서 제거해야 하는지 voting을 하게 되는데 이 단계가 reconfiguration입니다.

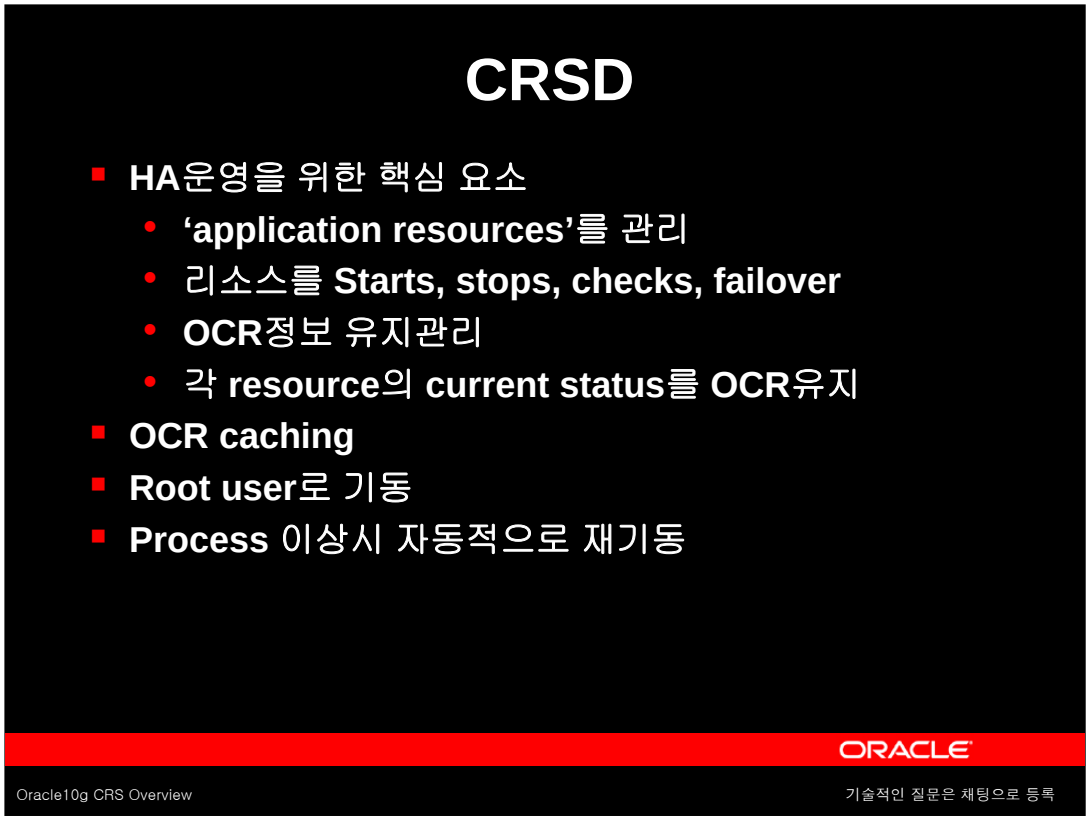
Nm이 어느 node가 죽었는지 판단하여 nm reconfiguration을 한 이후에, gm이 해당 node의 crs, evm, css등이 사용했던 resource들을 제거하고 cluster group에서 삭제하는 gm reconfiguration이 진행됩니다.

Reconfiguration은 node join시에도 발생합니다.



다음은 crsd에 대해 알아보겠습니다.

CRSD는 cluster내에서 사용하는 resource를 monitoring하고 관리하는 역할을 합니다.

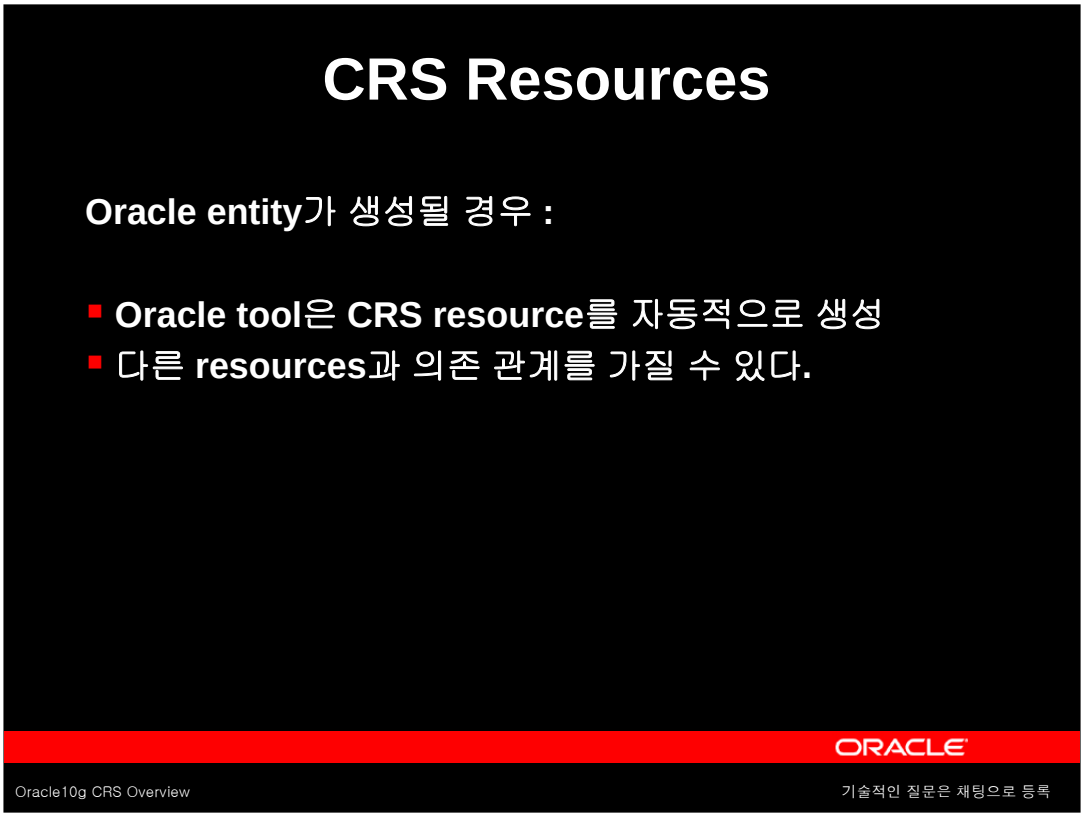


CRSD는 shell script인 racgwrap을 call하게 되고, 이 racgwrap은 racgmain을 call하여 특정 resource를 start/stop/check할 수 있습니다.

이 resource들을 관리하기 위해 해당 resource의 정보를 담아야 하는 공간이 필요하며, 이것이 ocr입니다. CRSD는 이 OCR내의 정보를 직접 read/write하여 data를 유지 관리하는 역할을 합니다.

Read/write시 disk io를 최소화하기위해 OCR data를 memory에 caching하는 기능도 제공합니다.

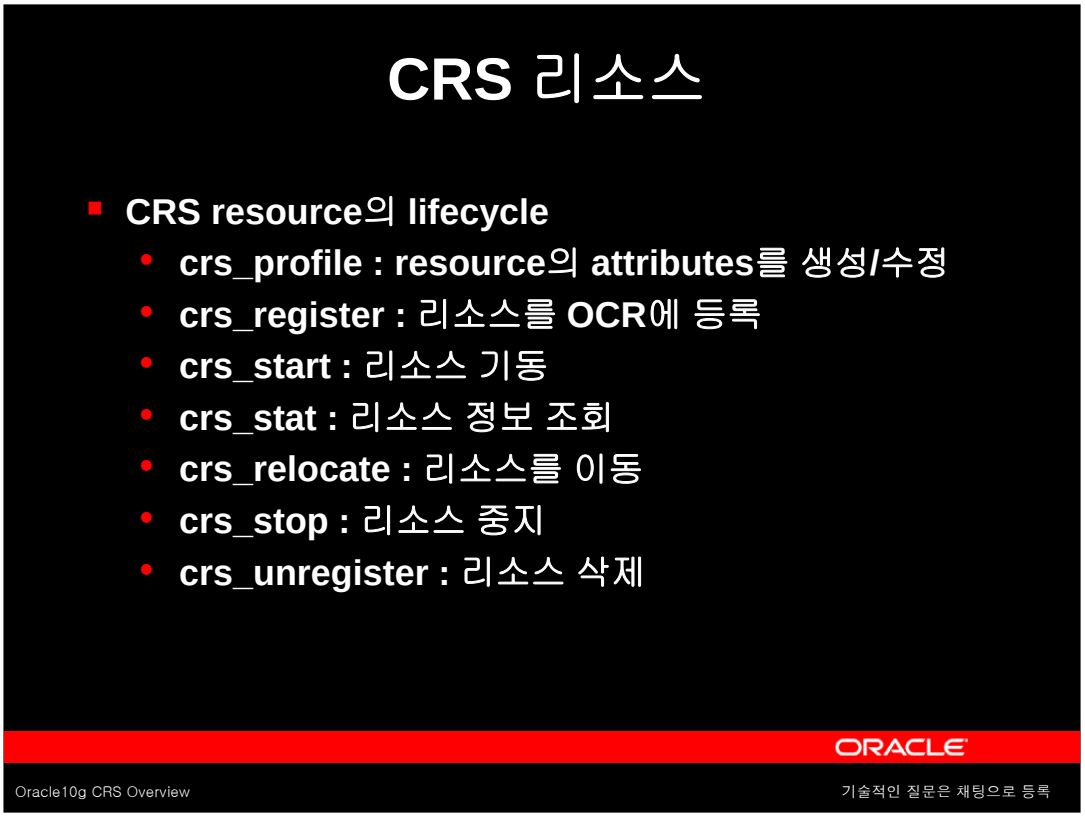
CRSD는 root user로 기동하고, process fail이 발생하면 자동적으로 재기동 됩니다.



Oracle entity는 database, instance, service나 listener등이 있습니다. 이런 oracle entity의 생성은 dbca, netca, srvctl등의 oracle tool들에 의해 생성하도록 요청되고, CRSD가 실질적으로 OCR을 update하여 생성하게 됩니다.

이런 crs resource들간에는 dependency를 가질 수 있습니다. 예를 들어 database instance나 listener는 vip가 있어야만 생성될 수 있고, start될 수 있습니다.

이 경우 dependency가 성립됩니다.



Oracle crs가 crs resource를 관리하기위해 내부적으로 관리하는 명령어들은 다음과 같습니다.

이 명령어들이 담당하는 역할은 하나의 resource의 lifecycle과 동일합니다.

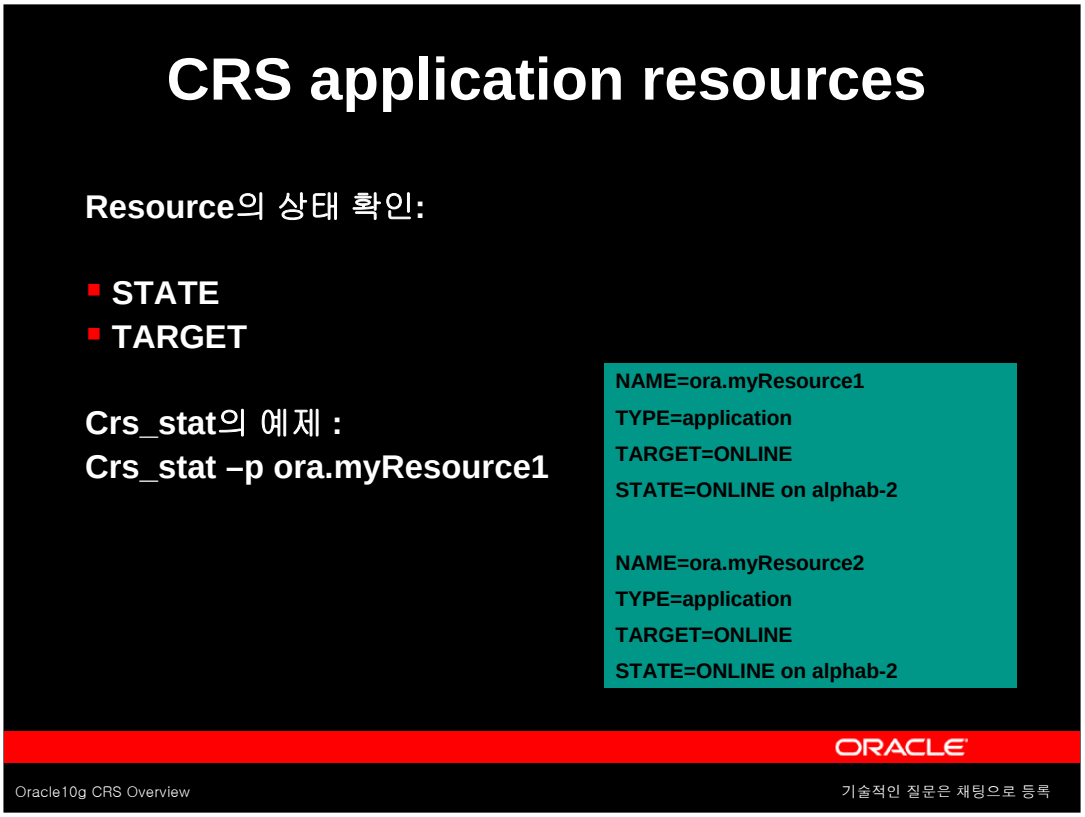
Crs_profile은 resource의 각종 attribute를 정의합니다.

일단 attribute가 정의되면, crs_register에 의해 resource로서 추가되고, ocr에 쓰여지게 됩니다.

Crs_start는 resource를 start할때 사용됩니다. Crs_stat는 해당 resource의 정보를 조회할때 사용합니다. Crs_relocate는 해당 resource를 move할때 사용합니다.

Crs-stop은 resource를 stop하며, crs_unregister를 이용하여 resource를 삭제할 수 있습니다.

Resource를 관리한다는 것은 위의 7가지 상태를 말하며, 이 7개의 단계는 resource의 life cycle이 됩니다.

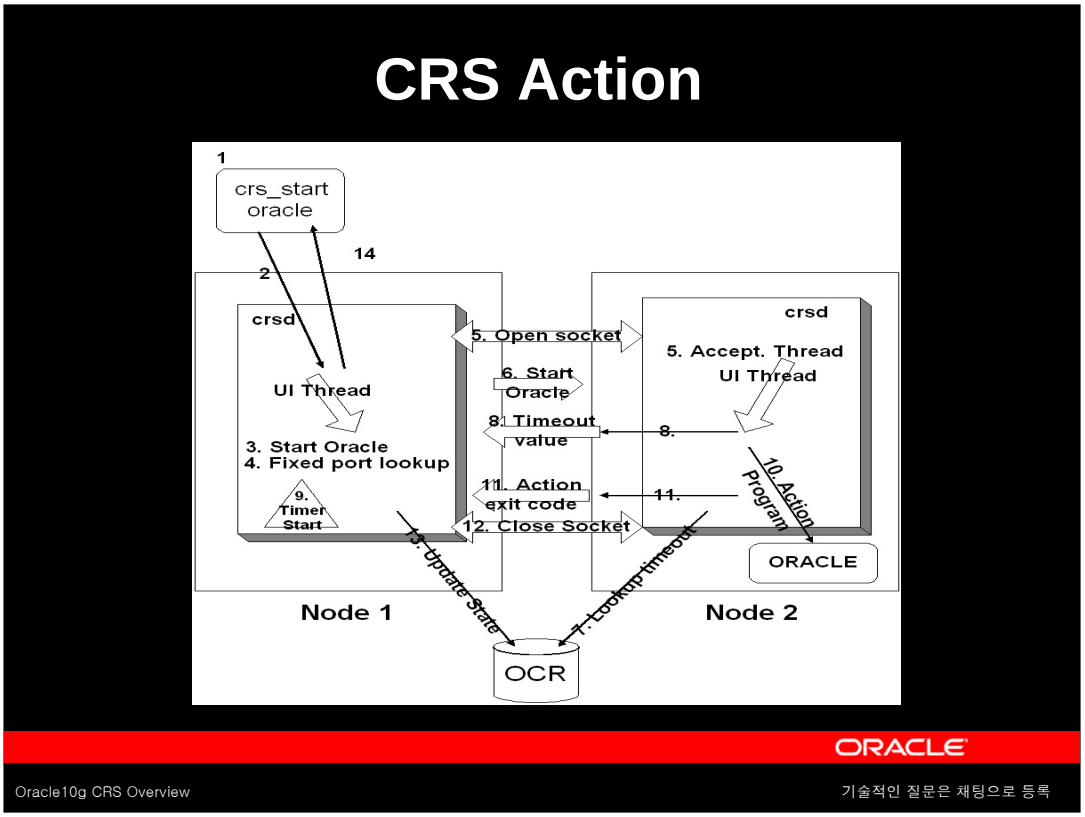


일단 resource가 startup되면 crsd는 resource마다 하나씩 thread를 생성하여 이 resource를 check_interval을 주기로 정상작동 여부를 check하게 됩니다.

Resource가 비정상 종료되면(예를 들어 instance crash) resource의 state는 offline으로 변경되고, restart_attemps에 해당되는 숫자만큼 restart를 자동적으로 시도하게 됩니다. 이 숫자를 넘어서게 되면 더이상 restart를 시도하지 않습니다.

Crs resource의 현재 상태는 crs_stat을 통하여 알 수 있습니다.

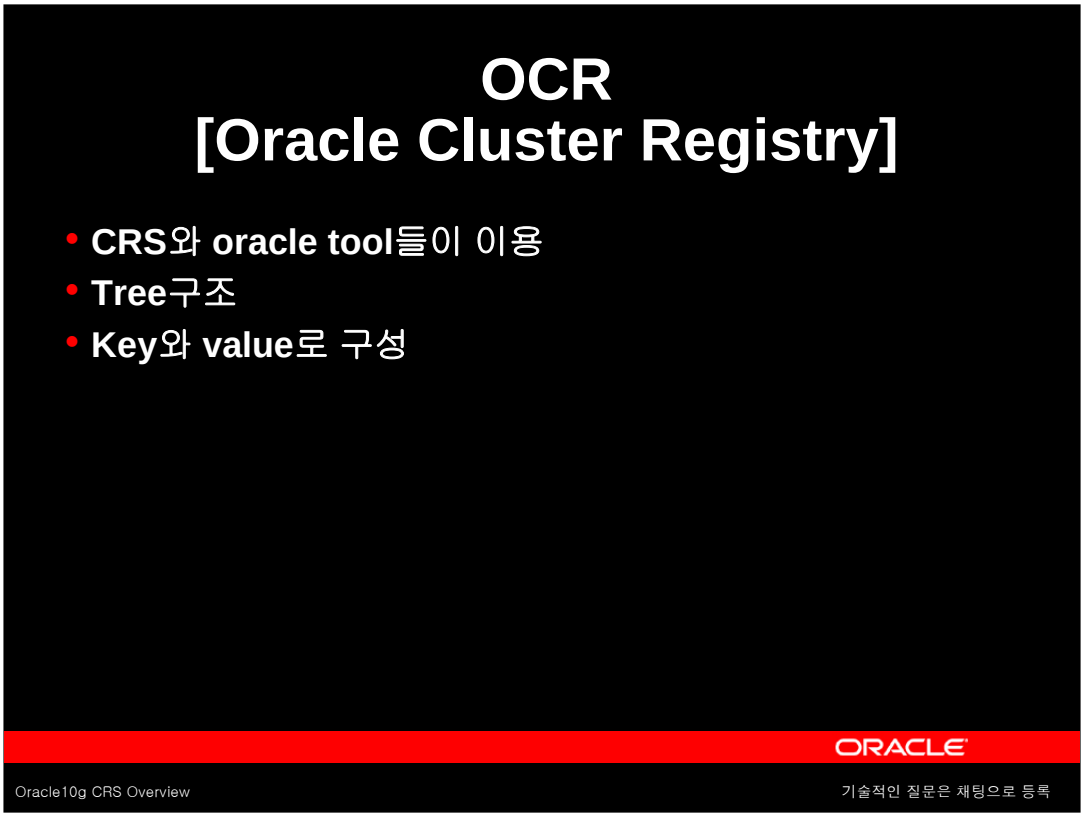
이 명령 결과로 조회되는 state는 current상태를 표시하고, target은 crs가 유지해야 하는 resource의 상태입니다. 즉, target이 대부분 online으로 되어있으므로 crs는 해당 resource의 current state를 online으로 유지하려고 합니다.



Crsd가 resource를 관리하기위해 node간에 message를 주고받는 예를 한번 보겠습니다.

Node는 2개이고, instnace start상황이라고 가정을 하겠습니다.

1. Node1의 client에서 api를 통해 'crs_start oracle'요청이 들어옵니다.
2. local CRSD는 이 요청을 받고, 이를 처리하기위해 새로운 thread를 생성합니다. 요청이 타당한지 검증을 하고, 어느 node에서 이 명령이 수행되어야 하는지 판단합니다. 여기서는 node2에서 작동하는 경우를 예로 들겠습니다.
3. Request=start resource=oracle로 key/value형태로 message를 생성합니다.
4. Node2의 crsd와 통신할 수 있는 port를 확인합니다.
5. Node2의 port가 open되고, socket이 open됩니다. Node2의 crsd는 이 요청을 처리하기위한 thread를 생성합니다.
6. 3번에서 생성한 message를 node2로 보냅니다.
7. Node2의 crsd는 message를 decode하고, 이 message가 타당한지 여부를 확인하기 위해 ocr을 확인합니다.특히, timeout value인 script_timeout의 값을 확인합니다.
8. Node2는 script_timeout의 값을 node1에 보내서 응답을 이 시간동안 기다리도록 합니다.
9. Node1은 timer를 시작하여 이 script_timeout동안 응답이 오기를 기다립니다.
10. Node2는 resource를 start합니다.
11. Node2는 node1으로 action program이 수행되었음을 socker을 통해서 보냅니다.
12. Node2에서 socket을 close합니다.
13. Node1에서 resource의 status를 변경함으로써 모든 작업이 완료됩니다.



OCR은 oracle cluster registry의 약자입니다. CRS에서 사용하는 data를 저장하는 repository역할을 합니다. Data의 구조는 window의 registry처럼 tree구조로 되어있고, key-value의 형태로 data가 저장되어있습니다.

이 OCR data는 DBCA, NetCA, SRVCTL등의 oracle tool과 CRS가 사용합니다.

Srvctl에서 resource를 생성하는 예를 보면,

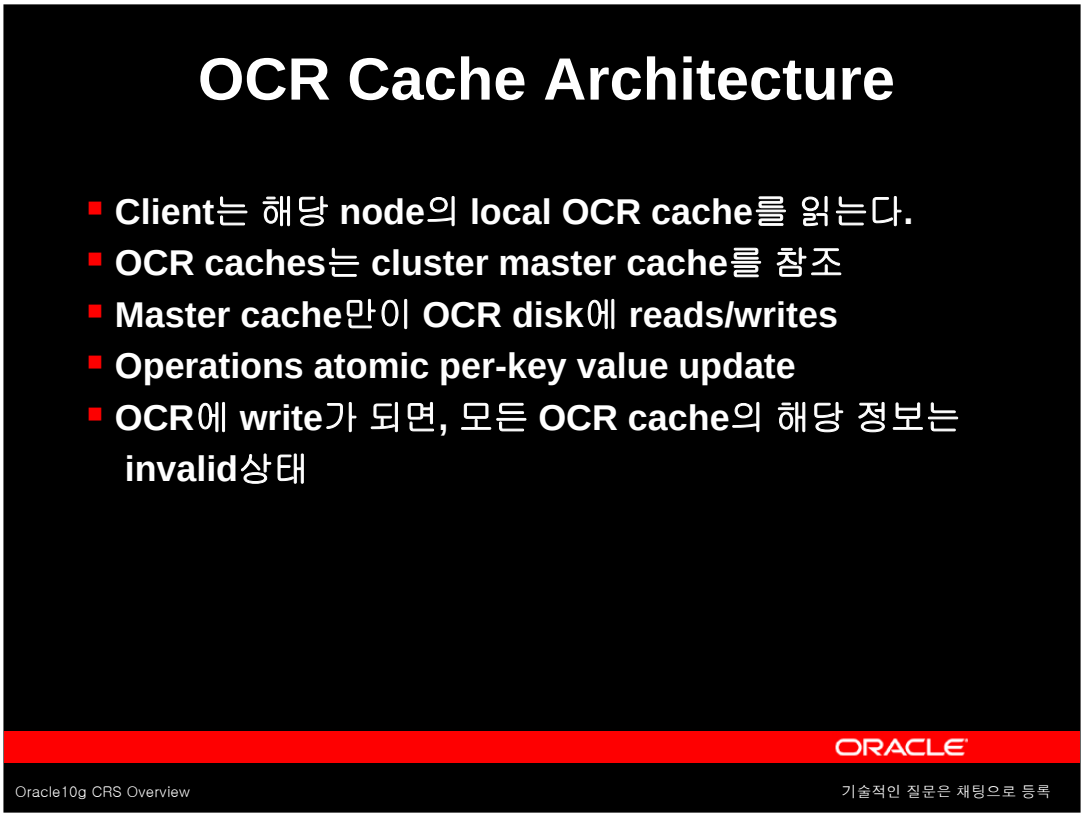
srvctl create database –d *mydatabase* –o *myOracleHome*

의 형태입니다.

이 명령으로 OCR에는 mydatabase라는 database가 추가됩니다. 그러나 실제로 database가 생성되는것은 아닙니다. Database의 생성은 'create database'문장으로 생성을 해야 합니다.

Dbca를 이용하면 database를 실제로 생성한 후 OCR에 등록하는 역할까지 합니다.

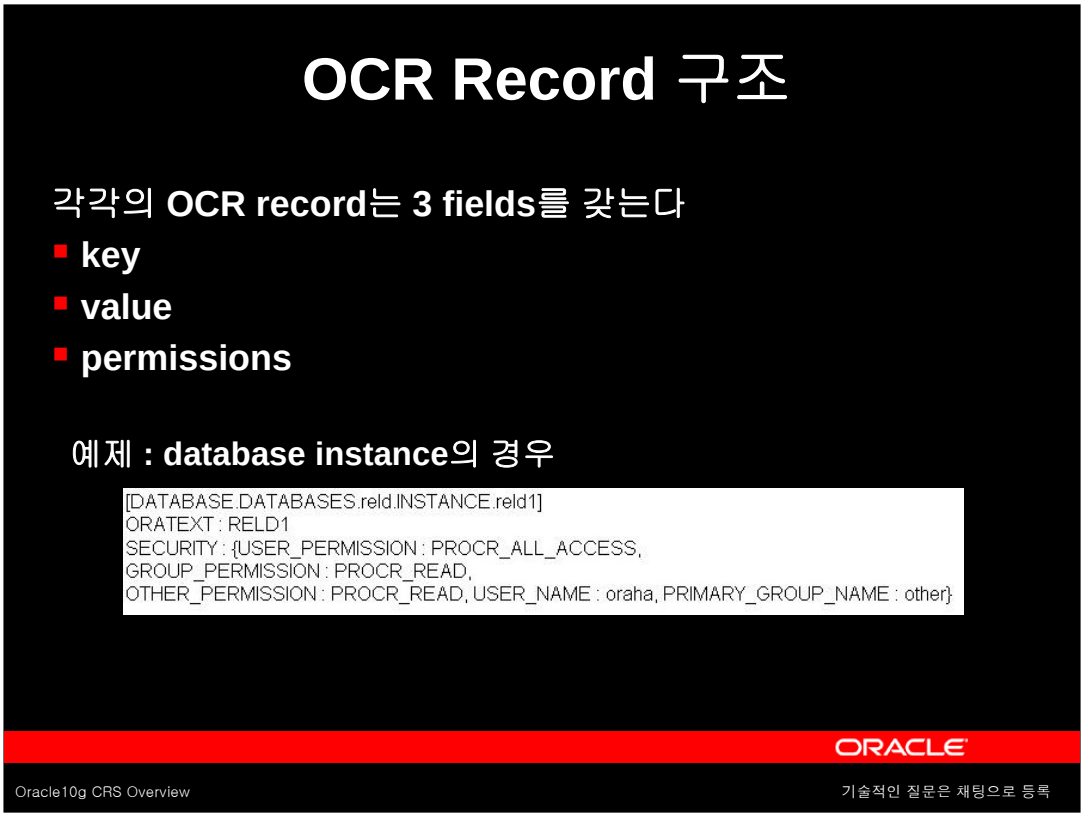
OCR은 shared device에 생성되어야 하며, raw device나 cluster file system도 가능합니다. OCR의 위치는 일반적으로 /var/opt/oracle/srvConfig.loc file에 지정되어있거나, 환경 변수인 SRV_CONFIG로 지정이 가능합니다.



Srvctl등의 oracle tool이 OCR정보를 access해야되는 경우 local node의 crsd가 관리하는 local ocr cache를 읽습니다. Ocr cache는 각 node의 crs에 의해 관리되지만 그중의 한 crs는 cluster master cache로서 역할을 합니다.

이 master cache만이 OCR disk에 disk io를 발생시키게 되고 read한 data를 다른 node의 ocr cache로 전파합니다.

Resouce start/add등 OCR disk에 write해야 되는 상황에도 master cache를 담당하는 crsd가 write를 하고, 이 사실을 다른 node의 crsd에 전파합니다. OCR write가 발생하면, 기존에 cache되어있던 해당 data는 invalid상태로 변경되어 master ocr cache로부터 다시 전달 받아야 합니다.



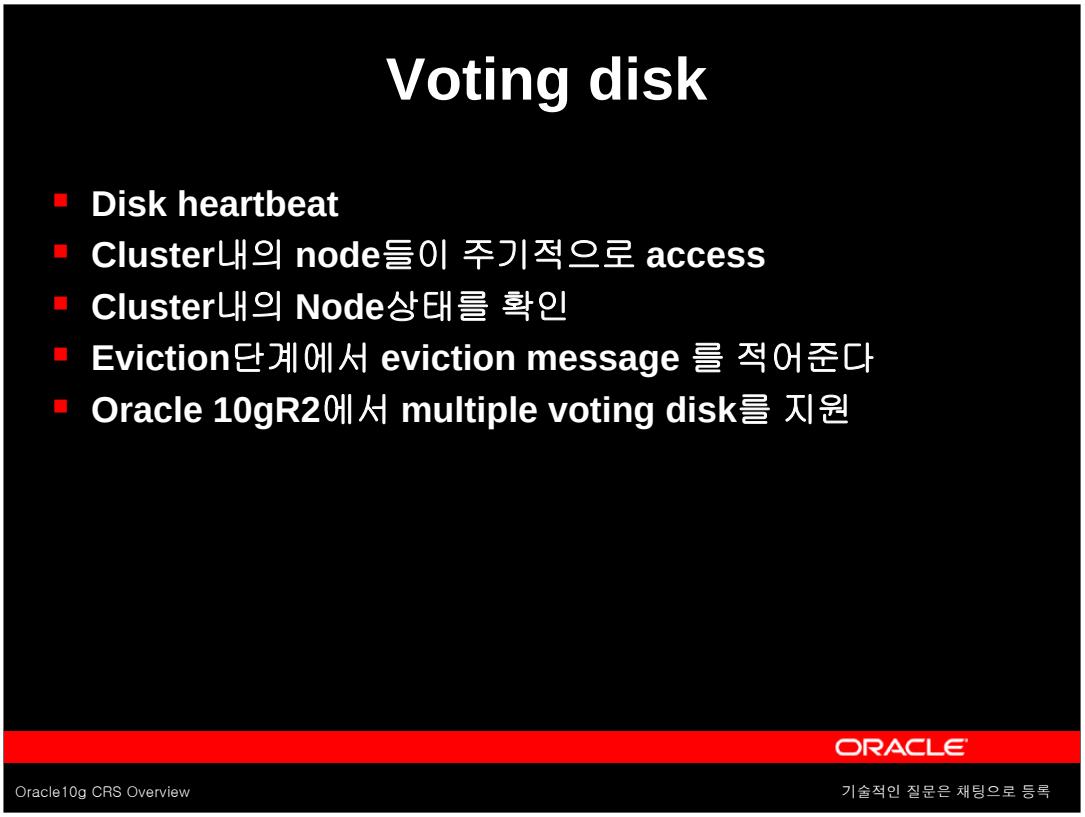
각 ocr record는 key, value, permission의 3개의 field를 갖습니다.

여기 예에서 보면, 제일 위의

Databases.reld.instance.reld1은 key입니다. 여기서는 database instance를 예를 들었습니다.

그 밑에 oratext항목에 나타나는 reld1은 key에 대한 value입니다. 여기서는 instance명인 reld1이 value입니다.

세번째 항목은 security입니다. Unix에서 permission을 관리하는 방식처럼 user, group, other group의 세 부분으로 나뉘고, username과 primary_group_name에 이 user의 user명과 그룹 명이 표시됩니다. 여기 예에서는 oraha user에 other group으로 표시되었지만, 일반적인 경우에 oracle user에 dba group이 표시됩니다.



Voting disk는 split brain상태에서 node의 상태를 판단하기위한 second heart beat의 역할을 합니다. Nm은 이 상태에서 어느 sub cluster node를 evict할지 결정하기 위해 voting disk를 사용합니다.

Eviction이 결정되면 해당 node가 eviction되도록 eviction message를 voting disk에 적어줍니다.

Voting disk는 dd로 backup을 받아 disk failure시 복구할 수 있습니다.

10gR1에서는 multiple voting disk를 지원하지 않으므로 os level의 mirroring 기능을 이용해야 하지만, 10gR2에서 multiple voting disk기능을 지원합니다.

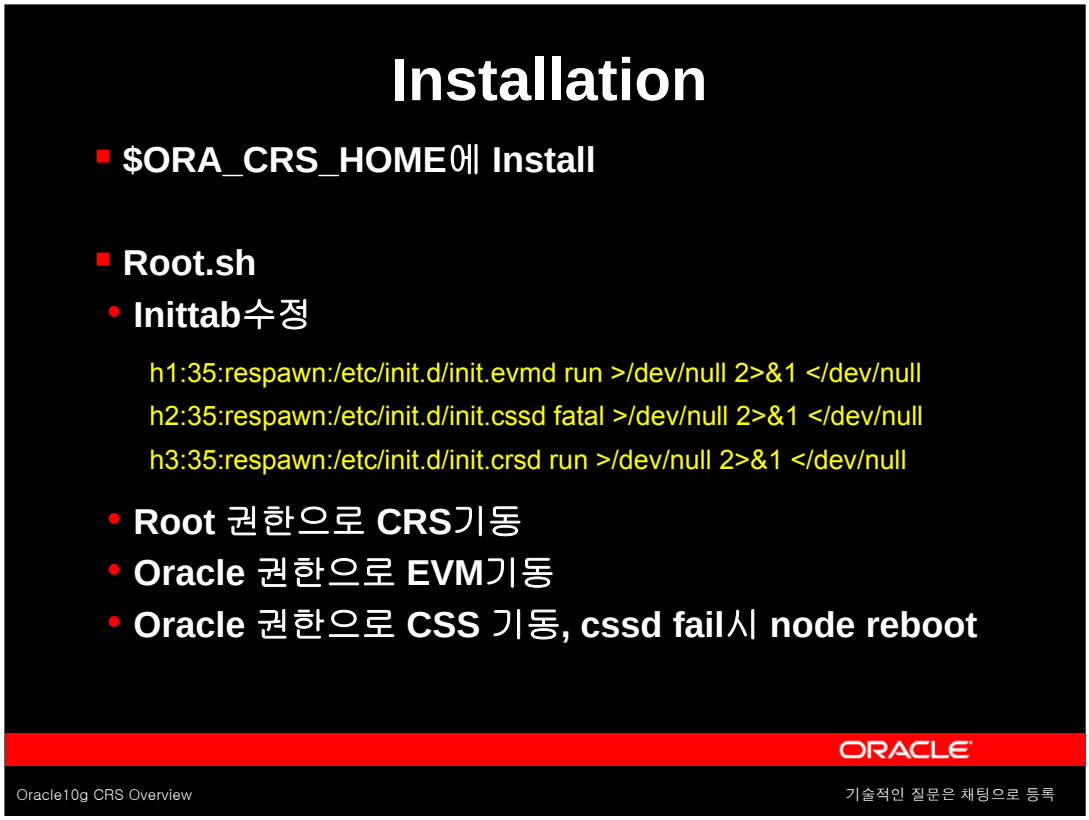
참고로 OCR backup은 $ORA_CRS_HOME/cdata/<cluster name>/ direcotry에 자동적으로 backup이 됩니다.

Backup은 매 4시간마다 1개, 매일 1개, 매주 1개의 갯수로 유지됩니다.

Oraconfig –showbackup 명령으로 가용한 backup을 알수 있으며,

oraconfig –restore filename 명령으로 restore할 수 있습니다. Restore시는 crs를 down후 restore해야 합니다.

자세한 절차는 metalink에서 note. 268937.1를 확인하시기바랍니다.

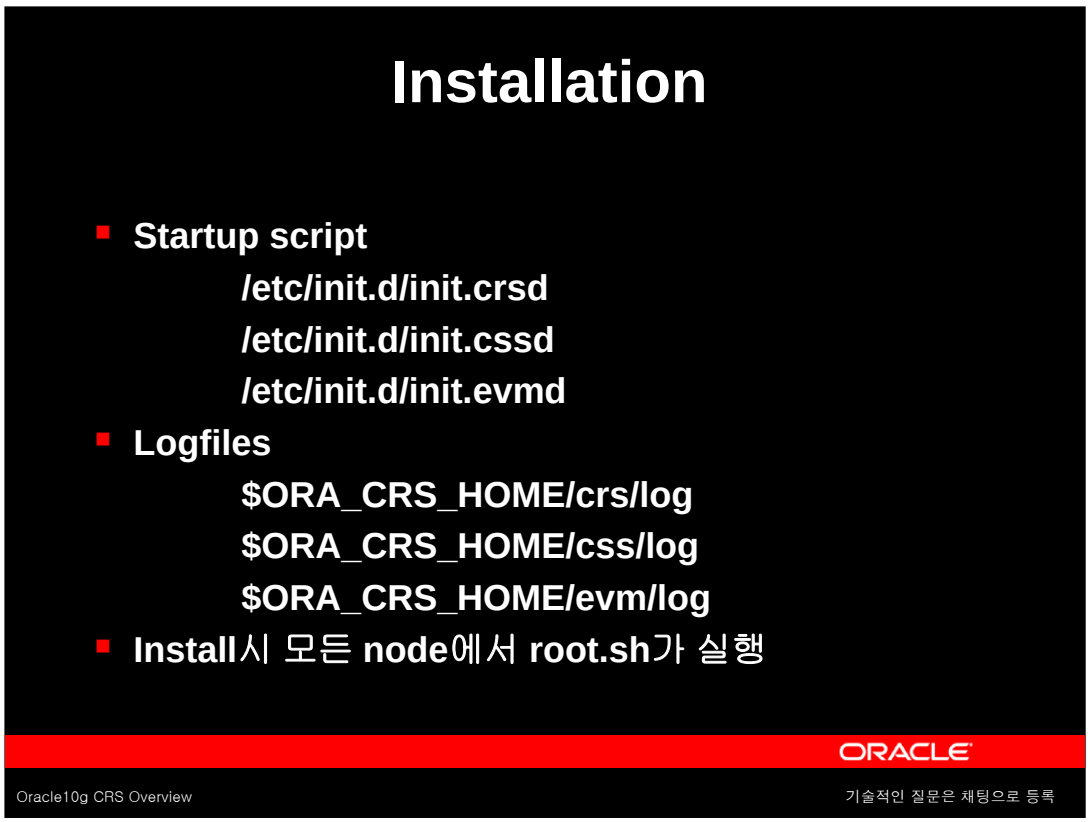


CRS는 $ORACLE_HOME과는 별도의 경로에 $ORA_CRS_HOME이라는 환경 변수를 설정 후, 이 directory에 install합니다.

Install과정의 마지막 부분에 root.sh shell을 실행하도록 message가 나타납니다. 이 root.sh는 inittab file에 crs process들이 booting시 startup되도록 inittab에 관련 정보를 기록합니다.

이 inittab들 보면 crs는 root권한으로 evm은 oracle권한으로 기동되며, process fail시 자동으로 재기동됩니다.

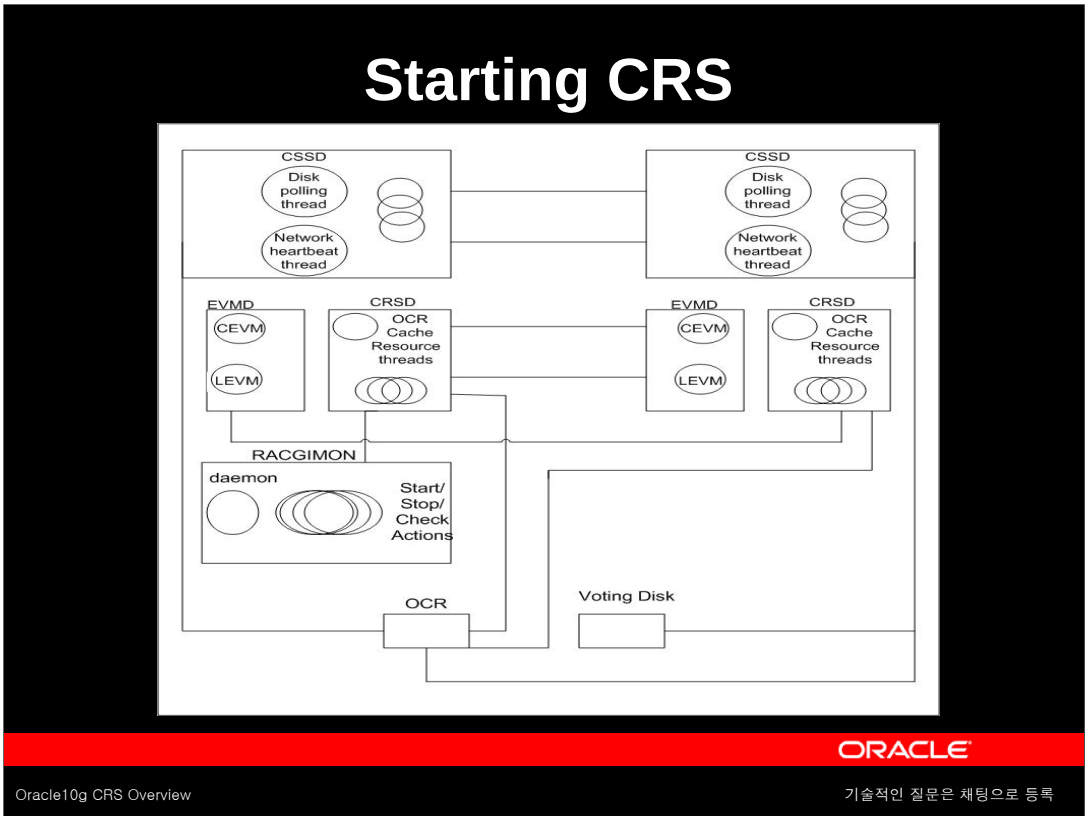
Css는 oracle권한으로 기동되며, fatal mode로 기동되므로 process fail시 os reboot이 됩니다.



Booting시 실행되는 script는 init.crsd, init.cssd, init.evmd 이 세개가 실행되어 crs가 기동되게 됩니다.

이 process들은 $ORA_CRS_HOME/<process명>/log directory에 각각의 log를 관리합니다.
따라서 각 process에 이상 발생시 우선적으로 이 log directory를 참조 하시기 바랍니다.

Install시 root.sh가 모든 node에서 실행되어야 합니다. 만약 정상적으로 실행되지 않은 경우라면, 문제발생시 원인을 찾기 힘들게 됩니다.



CRS가 start되는 과정을 다시 한번 review해보겠습니다.

Init.crsd, init.evmd, init.cssd는 init.crs에 의해 call이되어, manual하게 start될 수 있습니다.

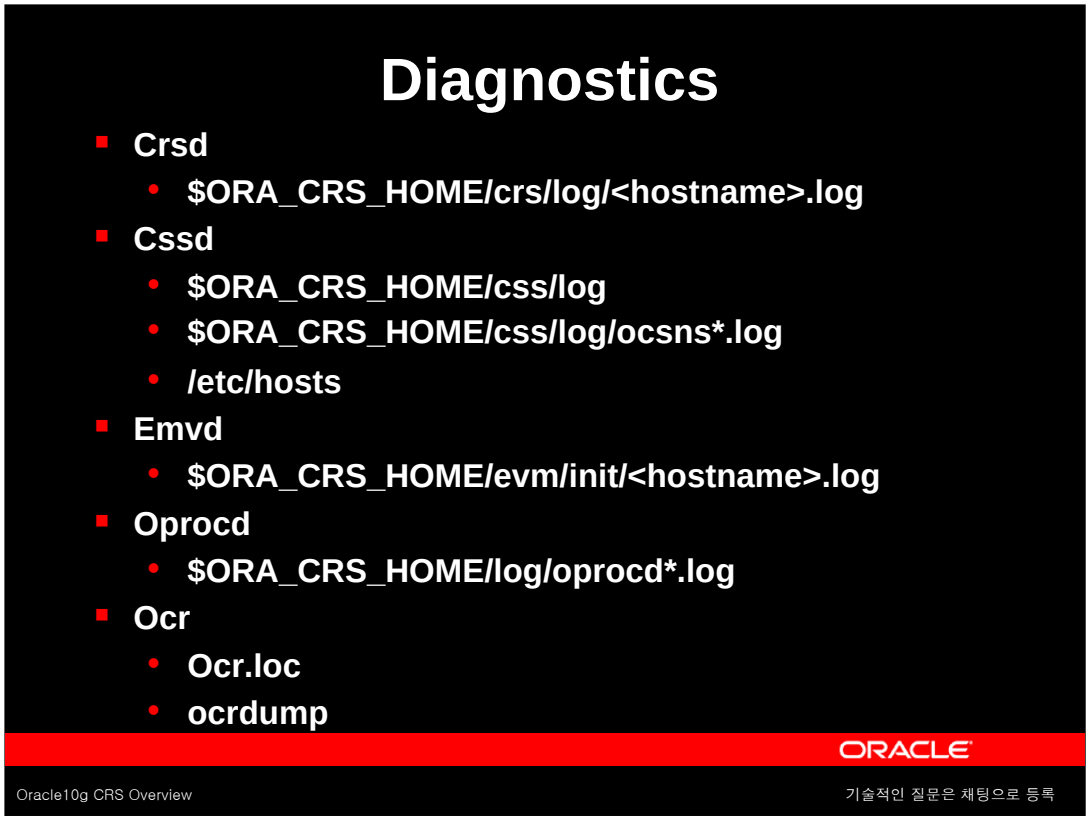
'Init.crs start'명령이 발생하면, 먼저 cssd를 start하여 cluster를 형성하게 되고, 이후 evm이 start됩니다. 마지막으로 crsd가 start되어 crs가 모두 기동됩니다.

Css의 주요 thread는 voting disk polling thread와 network heartbeat thread입니다. Disk polling thread는 매초마다 voting disk를 read합니다. Heartbeat thread도 또한 매초마다 작업합니다. Css는 이외에 timer thread, 기타 관리를 위한 thread등 이 있습니다.

Css는 startup하면서 OCR과 voting disk를 읽어 cluster node들을 확인합니다. 이단계에서 OCR이나 Voting disk를 read할 수 없는 상황이 되면 start되지 못합니다. 이후 evm이 start됩니다. CRSD가 start되면서 OCR data를 cache하기 위해 OCR cache thread를 기동합니다. OCR cache는 node간에 master-slave 관계를 갖게 되는데, 가장 먼저 booting된 node가 master가 되고, OCR disk에 대한 write는 이 master에 의해서만 이루어집니다. 이후 변경된 data의 전파도 이 master crs가 하게 됩니다.

Crsd는 ocr을 읽어서 모든 resource들을 파악하고, 이들간의 dependency도 확인합니다. 이후 racgwrap을 call하여 각각의 resource들을 start합니다. Racgwrap은 racgmail을 call하면서 argument로써 resource name과 start를 넘겨줍니다. 각각의 resource의 health check를 하기위해 매 초마다 sga의 특정영역을 polling합니다.

이 단계를 마치게 되면 모든 crs daemon들과 nodeapps, database, instance등의 resource가 모든 node에서 기동된 상태가 됩니다.



다음은 문제 초기단계에서 볼 수 있는 trace file들을 확인해보겠습니다.

Crs, css, evm은 Oracle10gR1의 경우는 $ORA_CRS_HOME directory밑에 process명 아래에 log directory에 해당 log가 있습니다.

우선적으로 봐야 할 log file은 이 log file들입니다.

Css의 경우는 interconnect의 networking이 안 되는 경우도 있으므로, css의 network trace file인 $ORA_CRS_HOME/css/log/ocsns*.log 가 필요할 수도 있습니다. 그리고 network configuration을 파악하기위해 hosts, netstat결과도 확인이 필요합니다.

Emvd의 log는 각각 다음과 같습니다.

`EVM daemon log는 $ORA_CRS_HOME/evm/init/<hostname>.log`

`EVM event logger log는 $ORA_CRS_HOME/evm/log/<hostname>_evmdaemon.log`

Oprocd가 있는 경우는 $ORA_CRS_HOME/log에 oprocd log file이 있습니다.

Ocr의 이상시는 먼저 ocr의 위치를 지정하고있는 ocr.loc file을 확인하세요. 이 file은 linux의 경우는 /etc/oracle, 기타 unix는 `/var/opt/oracle` 에 위치합니다. Ocr내부의 data를 확인하기위해 ocrdump 명령을 사용할 수 있습니다. 별도의 이름을 주지 않은 경우 OCRDUMP라는 file명으로 생성됩니다.

공통적으로 OS system log를 확인하여, network error나 io error, 혹은 다른 message가 있는지 확인이 필요합니다.

CRS의 경우도 여러 형태의 문제가 발생할 수 있기 때문에 문제유형별로 trace를 발생시킬 수 있습니다. 이 내용에 대해서는 iSeminar에서 언급하지는 않겠습니다. 위의 기본적인 log file들과 함께 Oracle Suppport Engineer와 함께 진행하시기 바랍니다.

참고문서

- Note. 272332.1
  CRS 10g Diagnostic Collection Guide
- Note. 259301.1
  CRS and 10g Real Application Clusters
- Note. 279793.1
  How to Restore a Lost Voting Disk in 10g
- Note. 268937.1
  Repairing or Restoring an Inconsistent OCR in RAC

ORACLE