• **Cluster:**
  - A cluster is a group of independent, but interconnected computers that acts as a single system.
  - Deployed to increase the availability, performance and to balance a dynamically changed workload or to balance the load across the systems.
  - Most cluster architectures use a dedicated interconnect network for communication and coordination between cluster nodes. It also uses shared disk storage architecture.

• **Clusterware:**
  - Clusterware is a software that provides various interfaces and services for a cluster. Typically, this includes capabilities that:
    - Allow the cluster to be managed as a single entity.
    - Protect the integrity of the cluster so that data is protected and the cluster continues to function even if communication with a cluster node is severed.
    - Maintain a registry of resources so that their location is known across the cluster and so that dependencies between resources is maintained.
    - Deal with changes to the cluster such as node additions, removals, or failures.
    - Provides a common view of resources such as network address and files in a file system

• **Oracle Clusterware:**
  - Oracle Clusterware provides above capabilities and it’s a key part of a Grid Infrastructure and integrated with Oracle Automatic Storage Management (ASM) and ASM Cluster File system (ACFS).
  - Combination of Oracle Clusterware, ASM and ACFS provides foundation for Oracle Real Application Clusters (RAC) database.
  - Services provided by oracle clusterware include:
    - **Cluster Management:** Allows cluster services and application resources to be monitored and managed from any node in cluster.
    - **Node Monitoring:** Provides real-time information regarding which nodes are currently available and the resources they support. Cluster integrity is also protected by evict/nf or fencing unresponsive nodes.
    - **Event Services:** Publishes cluster events so that applications are aware of changes in the cluster.
    - **Time Synchronization:** Synchronizes the time on all nodes of the cluster.
    - **Network Management:** Provisions and manages Virtual IP (VIP) addresses that are associated with cluster nodes to provide a consistent network identity regardless of which nodes are available. In addition, Grid Naming Service (GNS) manages network naming within the cluster.
    - **High Availability:** Services, monitors and restarts all other resources as required.
Oracle Clusterware Startup:
- Oracle clusterware is started by the OS init daemon.
- Oracle clusterware installation modifies the `/etc/inittab` file to restart `ohasd` in the event of a crash.
- Oracle High Availability Services daemon (`ohasd`) is responsible for starting in proper order, monitoring, and restarting other local oracle daemons, up through the `crsd` daemon, which manages clusterwide resources.
Oracle Clusterware Process:

- CRS (CRS daemon):
  - Manages cluster resources based on configuration information that is stored in Oracle Cluster Registry (OCR) for each resource. This includes start, stop, monitor and failover operations. crsd generates events when the status of a resource changes.
  - When a resource fails it will try to restart and if configured will failover to another node after exhausting restart attempts.
  - It is responsible to maintain the OCR.
  - CRSD process writes to trace files in $GRID_HOME/log/<node-name>/crs/crsd.log

Cluster synchronization Service:

- Manages the cluster configuration by controlling which nodes are members of the cluster and by notifying members when a node joins or leaves the cluster.
- It master the first active node and other nodes in the cluster are synchronized by registering with master node.
- It writes trace file to in $GRID_HOME/log/<node-name>/css/cssd.log
- CSS has three separate process: the CSS daemon(ocssd), the CSS Agent(cssdagent), and the CSS Monitor(cssdmonitor).
  1. cssdmonitor => Monitors node hangs (via oprocd functionality) and monitors OCCSD process hangs (via oclsmom functionality) and monitors vendor clusterware(via vmon functionality). This is the multi threaded process that runs with elevated priority.
     Startup sequence: INIT --> init.ohasd --> ohasd --> ohasd.bin --> ccssdmonitor
  2. cssdagent => Spawned by OHASD process. Previously(10g) oprocd, responsible for I/O fencing. Killing this process would cause node reboot. Stops, start checks the status of ocssd.bin daemon
     Startup sequence: INIT --> init.ohasd --> ohasd --> ohasd.bin --> cssdagent
  3. ocssd.bin => Manages cluster node membership runs as oragrid user. Failure of this process results in node restart.
     Startup sequence: INIT --> init.ohasd --> ohasd --> ohasd.bin --> cssdagent --> ocssd --> ocssd.bin

Event Manager(EVM):

- Process that publishes Oracle Clusterware events. It has two process evmd and evmlogger.
  1. evmd.bin => Distributes and communicates some cluster events to all of the cluster members so that they are aware of the cluster changes.
  2. evmlogger.bin => Started by EVMD.bin reads the configuration files and determines what events to subscribe to from EVMD and it runs user defined actions for those events.
- **Disk Monitor (Disk Monitor Daemon):**
  - Monitors and performs i/o fencing for Oracle Exadata storage server. Diskmon daemon is always started when ocssd is started.

- **Multicast Domain name service (mDNS):**
  - Used by Grid Plug and Play to locate profiles in the cluster, as well as by GNS to perform name resolution. The mDNS process is a background process on Linux and UNIX and on Windows.

- **Grid Naming Service (Grid Naming Service Daemon):**
  - Is a gateway between the cluster mDNS and external DNS servers. The GNS process performs name resolution within the cluster.

- **Oracle Notification Service (ONS):**
  - It’s a publish and subscribe service for communication FAN events.

- **Oraagent:**
  - Extends clusterware to support Oracle-specific requirements and complex resources. This process runs server callout scripts when FAN events occur. This process was known as RACG in Oracle Clusterware 11g Release 1 (11.1).

  - **ohasd’s oraagent:**
    - Performs start/stop/check/clean actions for ora.asm, ora.evmd, ora.gipcd, ora.gpnpd, ora.mdnsd.

  - **Crsd’s oraagent:**
    - Performs start/stop/check/clean actions for ora.eons, ora.ons, SCAN listeners and listener.
    - Performs start/stop/check/clean for service, database and diskgroup resources.
    - Receives eONS events, and translates and forwards them to interested clients.

- **Cluster kill daemon (oclskd):**
  - Handles instance/node evictions requests that have been escalated to CSS.

- **Grid IPC daemon (gipcd):**
  - Is a helper daemon for the communication infrastructure.
- **Oraroot agent:**
  - A specialized oraagent process that helps crsd manages resources owned by root such as the network, and the grid virtual ip address.

- **Cluster Time Synchronization service (CTSS):**
  - Provides Time Management in a cluster for Oracle Clusterware

- **Grid Plug and Play (GPNDP):**
  - Provides access to the Grid Plug and Play profile, and coordinates updates to the profile among the nodes of the cluster to ensure that all of the nodes have the most recent profile.

- **System Monitor (Osysmond):**
  - The monitoring and operating system metric collection service that sends the data to cluster looger service. The service runs on every node in a cluster.

- **Clusterware Components:**
  - **Private Interconnect:**
    - used for communication between the instances. (for transferring of blocks across the instances)
    - It is recommended to isolate private interconnect to use different switch as it requires fast data transfer
    - private interconnect must be on different subnet than public and virual ip.
  - **Virual IP:**
    - Each Database node in RAC environment has one node ip and one node VIP address. The main difference between these two is node vips can move to any other system in case if current working system id down but node IP can't do that. Whenever client initiates connection using SCAN name scan listener replies with the lease loaded node-VIP.
  - **Voting disk:**
    - Voting disk is used for polling mechanism where each and every node in the cluster reports is health by polling the disk.
    - It also arbitrates cluster ownership among the instances in case of network failures.
    - To get votedisk location:
      ```
      $[crsctl query css votedisk.
      ```
OCR (Oracle Cluster Registry):

- OCR is used to maintain cluster resources as it stores the information about cluster resources as well as the resources defined in the cluster.
- Registry is a file stored in shared storage.
- Each node maintains a copy of OCR in memory only one process is designated as master process.
- Read from the OCR is performed by master OCR process and refresh the local copies where as write to OCR is performed by the local OCR process.
- CRSD master automatically creates OCR backups every four hours, and CRS retains the last three copies. The CRSD process also creates an OCR backup at the beginning of each day and of each week, and retains the last two copies.
- OCRCHECK: -- Displays health of OCR (Oracle Cluster Registry).
- To get the information about OCR disks
  $ ocrcheck
- To get the information about OLR(11gR2)
  $ ocrcheck -local -config
- OLR resides on every node in the cluster and manages Oracle Clusterware configuration information for each particular node
- To list ocr backup's
  $ ocrconfig -showbackup---lists all auto and manual backup
- To take manual backup
  $ ocrconfig -manualbackup---in 11g
- To take an export of ocr file
  $ ocrconfig -export <dumpfile>
- To restore OCR from backup
  $ ocrconfig -restore <backupfile>
  $ ocrconfig-import <dumpfile>
- To change the autobackup location
  $ ocrconfig -backuploc <newlocation>
SCAN:

- Single Client Access Name (SCAN) is a new Oracle Real Application Clusters (RAC) 11g Release 2 feature that provides
- a single name for clients to access an Oracle Database running in a cluster.
- The benefit is clients using SCAN do not need to change if you add or remove nodes in the cluster. Having a single name to access the cluster allows clients to use the EZConnect client and the simple JDBC thin URL to access any database running in the clusters independently of which server(s) in the cluster the database is active.
- SCAN provides load balancing and failover of client connections to the database.
- The SCAN works as an IP alias for the cluster.
- SCAN is tied to maximum of 3 ipaddress and it can be configured in DNS or GNS. It works in roundrobin.
- Each SCAN IP(SCAN VIP) has a SCAN LISTENER associated with it. If it’s a 4 node RAC cluster 3 SCAN LISTENERS running on 3 SCAN ips’ will run on 3 nodes.
- All databases are registered with each SCAN LISTENER in the Cluster and PMON updates it's load to each SCAN LISTENER. Each request go through using SCAN_NAME, resolves to SCAN VIP i.e. SCAN LISTENER. Now, SCAN LISTENER redirects it to VIP by deciding using Load Balance.
- **Shared Storage:**
  - Datafiles, controlfiles, redologfiles, spfile, OCR and Voting disk must be present in the shared storage.
  - From 11g ASM is the go to filesystem for shared storage.

- **GPnP Profile:**
  - GPnP Profile is an XML file which contains information about cluster name, Network classification, storage to be used for ASM and digital signature of the cluster.

- How CSSD, CRS and ASM comes up if OCR and Voting disk is stored in ASM diskgroup:

```
1. Get ASM disk string
2. Do ASM disk discovery to locate vote disk
3. Once Vote disk located, startup CSS
4. ASM mounts DATA dg
```

---

**Diagram:**

- CSS
- GPnP
- OHASD
- CRS
- DATA DG
- SPFILE
- Vote
- OCR
- ASM Disk

Agents start GPnP, CSS and ASM

---

**Chart:**

- CSS
- GPnP
- OHASD
- CRM
- DATA DG
- SPFILE
- Vote
- OCR
- ASM Disk

Agents start GPnP, CSS and ASM
• **RAC Database:**
  - **Shared Resources:**
    - Datafiles
    - Control files
    - Temp tablespaces
    - Voting disk and OCR
    - Rac Interconnect
    - VIP's
  - **Non-shared resources:**
    - Online redolog files
    - Undo Tablespace
    - Archivelog files
    - SGA
    - Background process
    - Alertlog, listener.ora, tnsnames.ora and pfile
    - Trace and audit files.
  - **Interconnect-dedicated:**
    - Internode communication
    - Heart beats
    - Data blocks.
  - **Cache Coherency:**
    - Maintaining consistency of data blocks in the buffer cache of multiple instances is called cache coherency.
  - **Cache Fusion:**
    - Cache fusion is a mechanism by which blocks are transferred from one instance to another instance via private interconnect.
• **RAC specific SGA component:**
  - GRD (Global Resource Directory)
    - It is part of the shared pool, Maintained by GES (Global Enque services) and GCS (Glocal Cache service)
    - whenever a block is transferred out of local cache to another instance’s cache GRD is updated
    - it holds the meta data about data blocks, that are available in DBBC
    - it holds information like
      1. SCN (system change number)
      2. DBI (data block identifier)
      3. location of the block
      4. mode of the block
        I. **null (N)** - Null mode is the least restrictive mode. It indicates no access rights. acts as a place holder.
        II. **shared(S)** - Shared mode indicate that database block is being read and not modified. However another session can read the data block
        III. **exclusive(E)** - Exclusive mode indicate exclusive access to block. Other resource cannot have write over this data block. However it can have consistent read on this datablock.
    - role of the block
      I. **local**-When a data block is first read into the instance from the disk it has a local role. Meaning that only 1 copy of data block exists in the cache. No other instance cache has a copy of this block.
      II. **Global**-Global role indicates that multiple copy of data block exists in clustered instance. For example a user connected to one of the instance request for a data block. If the block is local in one instance and requested by another instance then block is transferred to the requesting node and the role will be updated as global.
  6. **types of datablock image**
    i. current image-update data block value
    ii. consistent image-previous data block value
    iii. past image-grd updated image
    - it convert to current image when instance is crash
• **RAC Database specific services:**
  - **Global Cache Service:**
    - Global Cache Service (GCS) is the main component of Oracle Cache Fusion technology. This is represented by background process LMSn. There can be max 10 LMS process for an instance. The main function of GCS is to track the status and location of data blocks. Status of data block means the mode and role of data block (I will explain mode and role further). GCS is the main mechanism by which cache coherency among “multiple cache” is maintained. GCS is also responsible for block transfer between the instances.
  - **Global Enqueue Service:**
    - The GES is primarily responsible for maintaining coherency in the dictionary and library caches. The dictionary cache consists of the data dictionary master information for each node in its SGA (System Global Area) primarily for quicker lookup and access. Any DDL committed from a requesting node needs to be sync’ed and written across all data dictionaries in all nodes of the RAC environment. The GES makes sure that the changes remain consistent across the nodes and that there are no discrepancies. Moreover, with the same directive, the locks must be created and maintained across the nodes and GES must ensure that there are no deadlocks across requesting nodes over access to the same objects. LMON, LCK and LMD processes work in tandem to make the GES operate in a smooth and seamless fashion.

• **RAC Database Background Process:**
  - **LMS(Global Cache Service Process):**
    - Maintains records of data file statuses and each cached block by recording information in a GRD.
    - Controls the flow of messages to remote instance and managed global data block access and transmit block images between buffer caches of different instances.
  - **LMON(Global Enqueue Service Monitor):**
    - Detects instance transitions (joins or leaves the cluster) and performs reconfiguration of GEC and GCS.
    - Monitors global enqueues and resources across the cluster and perform global enqueue recovery operations
    - Responsible for executing Dynamic local remastering every 10 mins.
LMD (Global Enqueue service Daemon):
- Processes incoming enque request messages and controls access to global resources.
- Performs distributed deadlocks detections.

LCK0 (Instance Enqueue Process):
- Manages non-cache fusion resource requests such as library and row cache requests.

LMHB:
- Monitors LMON, LMD and LMSn process to ensure they are running normally without blocking or spinning.
• **Dynamic Resource reconfiguration:**
  
  - When one instance departs the cluster, the GRD portion of that instance needs to be redistributed to the surviving nodes. Similarly, when a new instance enters the cluster, the GRD portions of the existing instances must be redistributed to create the GRD portion of the new instance. This is called dynamic resource reconfiguration.

• **Dynamic Resource remastering:**
  
  - In RAC every data block is mastered by an instance. Mastering a block simply means that master instance keeps track of the state of the block until the next configuration event occurs. GCS keeps track of the number of GC requests on a per-instance and per-object basis. This means that if an instance, compared to another is heavily accessing blocks from the same objects, the GCS can take the decision to dynamically migrate all of that object’s resources to the instance that is accessing the object most and making it master of those blocks. LMON, LMD and LMS processes are responsible for dynamic remastering.

  - Remastering can be triggered as a result of:

    - Manual remastering
    - Resource affinity (object affinity, undo affinity and files affinity)
    - Instance crash.

  - To find the current master and previous master of a resource:

    SQL> select o.object_name, m.CURRENT_MASTER,
              m.PREVIOUS_MASTER, m.REMASTER_CNT
          from dba_objects o, v$gcspfmaster_info m
          where o.data_object_id = 73181
          and m.data_object_id = 73181 ;

    SQL> select kj.kjblname, kj.kjblname2, kj.kjblowner,
              kj.kjblmaster
          from (select kjblname, kjblname2, kjblowner,
                kjblmaster, kjbllockp
                from x$kjb1
                where kjblname = '0x97'][0x4],[BL]' ) kj, x$le le
          where le.le_kjbl = kj.kjbllockp
          order by le.le_addr;
• **Working of cache fusion:**

<table>
<thead>
<tr>
<th>INSTANCE1</th>
<th>INSTANCE2</th>
<th>INSTANCE3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock held: none</td>
<td>Lock held: none</td>
<td>Lock held: SL0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shared Local</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No past image</td>
</tr>
<tr>
<td>Lock held: none</td>
<td>Lock held: SL0</td>
<td>Lock held: SL0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Shared Local</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No past image</td>
</tr>
<tr>
<td>Lock held: none</td>
<td>Lock held: XLO</td>
<td>Lock held: none</td>
</tr>
<tr>
<td></td>
<td></td>
<td>-------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Excl Local</td>
</tr>
<tr>
<td></td>
<td></td>
<td>No past image</td>
</tr>
<tr>
<td>Lock held: XGO</td>
<td>Lock held: NG1</td>
<td>Lock held: none</td>
</tr>
<tr>
<td></td>
<td></td>
<td>--------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Null Global</td>
</tr>
<tr>
<td></td>
<td></td>
<td>With 1 past image</td>
</tr>
</tbody>
</table>

In stage 1 datablock is requested by a user C who is connected to instance 3. So a datablock is read into the buffer cache of instance 3.

```
SQL> select sal from emp where empno=10; \rightarrow returns 300
```

In stage 2 same datablock is requested by a user B who is connected to instance 2. As the block is already present in instance 3 it will be shipped to instance 2 via private interconnect using cache fusion.

```
SQL> select sal from emp where empno=10; \rightarrow returns 300
```

In stage 3 user B who is connected to instance 2 tries to update the sal of employee for empno 10. At this stage, instance 2 acquires EXCLUSIVE lock for updating the data at instance 2 and SHARED lock from instance 3 is downgraded to NULL lock.role of the block is still local as it is dirtied for the first time only on instance 2

```
SQL> update emp set sal=400 where empno=10; commit;
```

In stage 4 user A who is connected to instance 1 tries to update the sal of employee for empno 10. Datablock is shipped from instance 2 to instance 1 and the lock on instance2 is downgraded to null and role is global and has 1 past image.instance1 will have xclusive local with no past image

```
SQL> update emp set sal=500 where empno=10;
```
In stage 5 user C connected to instance 3 tried to select the sal of empno 10. Datablock will be copied from instance 1 to instance 3 and the lock on instance 1 and instance 3 will be shared as instance 3 requested shared lock. If instance 3 requested exclusive lock then on instance 3 lock will be null.

SQL> select sal from emp where empno=10; → returns 500

In stage 6 User B issues the same select statement against the emp table with empno 10 on instance 2. Instance 2 will request for a consistent copy of buffer from another instance, which happens to be the current master. Therefore instance 1 will ship the block to instance 2, where it will be required with SG1 (SHARED, GLOBAL with 1 PAST IMAGE). So instance 2 mode becomes SG1.

In stage 7 User C on instance 3 updates the same row. Therefore the instance 3 requires an exclusive lock and instance 1 and instance 2 will be downgraded to NULL lock with GLOBAL role and 1 PAST IMAGE. Instance 3 will have EXCLUSIVE lock, GLOBAL role and with no PAST IMAGES (XG0).

In stage 8 The checkpoint is initiated and a “Write to Disk” takes place at instance 3. As a result previous past images will be discarded (as they are not required for recovery) and instance 3 will hold that block in EXCLUSIVE lock LOCAL role with no PAST IMAGES (XLO).
• RAC Administration:
  
  To troubleshoot cluster startup/cluster health issues:
  
  - Review the alertlog of Grid infrastructure under $GRID_HOME/log/<node-name>/alert<node>.log
  - Review the cluster background process specific logfiles under $GRID_HOME/log/<node-name>/<process>/ <process-name>.log
  - Review the ocrcheck logfile under $GRID_HOME/log/<node-name>/client/ocrcheck_<timestamp>.log file
  - Node evictions issues might include some additional clusterware logfiles.
    /etc/oracle/lastgasp/* or /var/opt/oracle/lastgasp/*
  - If the above logfiles doesn't provide sufficient information to troubleshoot the issue we can enable the trace/ debug settings to get the detail information. Use below commands to set the debug/trace levels:
    $]crsctl get log css/crs/evm/all- get the current level of the trace
    $]crsctl set log crs crsmain=4
    $]crsctl set log crs all-=3- you can set the level from 1 to 5 higher the value higher the details provided.
    $]export SRVM_TRACE=TRUE
  
  - CVU(Cluster Verification utility is used to collect pre and post cluster configuration details at various levels and various components. It also provides the ability to verify the cluster health.
    $]cluvfy comp healthcheck –collect cluster|database
    $]cluvfy comp nodecon –n <node1>,<node2>--to verify network connections
    $] cluvfy stage -pre crsinst –n <node1>,<node2>---to verify prechecks
  
  - Cluster Diagnostic Collection Tool(diagcollection.sh): As cluster manages several logfiles it may be time consuming to collect the files manually. Diagcollection utility refers various cluster log files and gathers required information to diagnose critical cluster problems
    $]diagcollection.sh –collect –crs $GRID_HOME
  
  - Log a ticket in my Oracle support to review the files and provide the cause for the issue.
  
  - Look at error report in AIX(errpt –a) or /var/log/messages(linux) or /var/adm/messages(Solaris) or /var/adm/syslog/syslog.log(HP-UX) to see any errors reported at the os level
• **Interconnect Troubleshooting:**
  - Look for lost blocks in AWR reports.
  - Look for errors in o/p of ifconfig –a
  - Look for packetreassembles in netstat -s

• **Node evictions:**
  - CSSD monitors nodes using 2 communication channels:
    - Private interconnect ↔ network Heartbeat
    - Voting Disk based communication ↔ Disk Heartbeat.
  - For N/W hearbeat nodes must respond in cssd_misscount time (default is 30 seconds)
  - Look for "heartbeat fatal" in cssd.log file.
  - N/W heartbeart failures result in node evictions.
  - For disk heartbeat each node in the cluster pings (r/w) the voting disks every second.
  - Nodes must receive response in (long/short) diskTimeout time.
  - Diskheartbeat failures will lead to node evictions search for "DskHeartbeat" in ocssl.log file.

• **Commands to manage Cluster resources:**
  - To find the master node in the cluster:
    - **Method#1)**
      - go to GRID_HOME/log/<node-name>/cssd
      - cat ocssl.log | grep -i 'master node' | tail -1---this will show us # of nodes in the cluster local node# and master node#
    - **Method#2)**
      - CRSD in Master node is resposible for performing auto backup of ocr, so looking at OCR backup we can identify the master node
      - ocrconfig -showbackup auto---List the node on which OCR was backed up
To check the CRS status on all the nodes:
   `crsctl check cluster --all`
   `crsctl status server --f`

To stop/start CRS on all the nodes (stops everything except ohasd)
   `crsctl stop/start cluster --all`

To stop/start crs on one node (including ohasd)
   `crsctl stop/start check crs`

To control scan listener:
   `srvctl stop/status/start scanListener`
   `srvctl start scanListener --i <scan_listener#> -n <node-name>`
   `srvctl stop scanListener --i <scan_listener#>`

To start/stop a specific cluster resource
   `crs_stop <resource-name>` → Need to run from the same node.
   `crs_start --c <node-name> <resource-name>` → if --c is used with start we can run it from other node.

To Manage ASM diskgroup (We cannot stop the diskgroup used by OCR and Voting disk)
   `srvctl start/stop/status diskgroup --g <dg-name> -node <node1>,<node2>`

We cannot stop the ASM alone (we need to bring down cluster)

To manage the local listener
   `srvctl status/stop/start listener --l <LISTENER-NAME> -n <node-name>`

To manage the database instance:
   `srvctl status/stop/start instance --i <instance-name> -d <db-name>`

To Manage database:
   `srvctl status/stop/start database --d <db-name> -o <open/mount/’read only’>`
   `srvctl config database --d <db-name>`
   `srvctl modify database --d <db-name> -o <ORACLE_HOME> -p <SPFILE-LOCATION>`
Useful clusterware commands:

- To check the nodes in the cluster:
  ```
  olsnodes
  ```

- To check the cluster name
  ```
  cemutlo –n
  ```

- To view subnets of the interfaces used and modify them
  ```
  oifcfg getif to get the interface subnet
  oifcfg setif –global eth1/<new-subnet>:<interface-type> to add new subnet
  ifconfig eth1 <new-priv-ip> netmask <value> broadcast <max value for the subnet>
  oifcfg delif –global eth1<old-subnet>:<interface-type> to old new subnet
  ```

- To check the voting disk
  ```
  crsctl query css votedisk
  ```

- To check the OCR integrity
  ```
  ocrcheck
  ```

- To move votedisk to different diskgroup
  ```
  crsctl replace votedisk <new-dg-name>
  ```

- To move OCR
  ```
  ocrconfig –manualbackup
  ocrconfig –add <new-dg-name>
  ocrcheck
  ocrconfig –delete <old-dg-name>
  ```

- To change the scan listener port
  ```
  srvctl modify listener –p <port#>
  ```
Global Cache Wait Events

- gc [current/cr] [multiblock] request
  - gc [current/cr] [2/3]-way
    - Received after two or three network hops, immediately after request
  - gc [current/cr] grant 2-way
    - Not received and not mastered locally. Grant received immediately
  - gc [current/cr] [block/grant] congested
    - Block or grant received with delay because of CPU or memory lack
  - gc [current/cr] block busy
    - Received but not sent immediately
  - gc current grant busy
    - Not received and not mastered locally. Grant received with delay
  - gc [current/cr] [failure/retry]
    - Not received because of failure
  - gc buffer busy
    - Block arrival time less than buffer pin time
2-way Block Request: (It will be on 2 node RAC)

- SGA1 sends a direct request to SGA2. So SGA1 waits on the gc current block request event.
- When SGA2 receives the request, its local LGWR process may need to flush some recovery information to its local redo log files.
- Then, SGA2 sends the requested block to SGA1. When the block arrives in SGA1, the wait event is complete, and is reflected as gc current block 2-way.
3-way block request

- This is a modified scenario for a cluster with more than two nodes. However, the master for this block is on a node that is different from that of the requestor, and where the block is cached.
- Thus, the request must be forwarded.
- When the block arrives in SGA1, the wait event is complete, and is reflected as gc current block 3-way.
2-way Grant

- In this scenario, a grant message is sent by the master because the requested block is not cached in any instance.
- If the local instance is the resource master, the grant happens immediately. If not, the grant is always 2-way, regardless of the number of instances in the cluster.
- The round-trip looks similar to a 2-way block round-trip, with the difference that the wire time is determined by a small message, and the processing does not involve the buffer cache.
RAC Installation Triage:

1. **System Provisioning**
   - **Upgrade?**
     - **YES**
       - Check pre-reqs runcluvfy.sh
       - **Pre-reqs Met?**
         - **YES**
           - Install
         - **NO**
           - Pre-reqs Met?
             - **YES**
               - Install
             - **NO**
               - Problem Before root.sh?
                 - **YES**
                   - 1056322.1
                 - **NO**
                   - 942166.1
     - **NO**
       - Pre-reqs Met?
         - **YES**
           - Install
         - **NO**
           - Problem Before root.sh?
             - **YES**
               - 1056322.1
             - **NO**
               - Problem Running rootupgrader.sh?
                 - **YES**
                   - 1364947.1
                 - **NO**
                   - Problem Running rootupgrade.sh?

2. **Engage appropriate team**
   - CVU Fixup Jobs
   - DBAs
   - Sysadmin
   - Networking
   - Storage
   - OS Vendor
   - HW Vendor
   - Oracle Support
   - Etc

3. **TFA Collector**
   - Success?
• **Why we need VIPS?**
  - Without VIPs if user tries to connect to RAC database with all the local ip's of the RAC in address list. If one of the node is down and if user connection will wait until it timesout before moving to other node. Timeout depends on OS and it may vary from sconds to 2 minutes.
  - With VIPS as CRS failover the VIP to other healthy node. When use tries to connect with VIPS it will get the no listener immediately and it moves to other node immediately.
  - In 11gR2 even with scan vip's it provides the same functionality.

• **Why SCAN?**
  - With SCAN whenever we add or delete a node user doesn’t need to make any changes to the connection string.