用 PXE 方法从裸机批量推 Oracle 11gR2 RAC 成套环境，并用 Shell 脚本在推出的 RAC 节点上批量部署 32 个 Oracle11gR2 RAC 备份恢复案例场景的方法

中国科学院 ARP 项目实施顾问，上海 Oracle 用户组成员 唐波

摘要

本文前半段介绍：用 PXE 推送端主机，把原先处于裸机状态的三台一套 PXE 被推送端主机批量推成 Oracle 11gR2 RAC 环境的方法。在每套推出的 Oracle 11gR2 RAC 环境中，都包含已自动安装并配置好的三台主机：一台共享磁盘主机和两台节点主机。每套推出的 Oracle 11gR2 RAC 环境中的两台节点主机都已自动挂接上该套环境内部的共享磁盘主机。两台节点主机各配备一个分布式虚拟磁带库，并且做好所有操作系统层面的配置：包括 vip、scan-vip、ntp，各种 rpm 包和操作系统参数等。

本文后半段介绍：如何复用以上的 PXE 推送端主机作为部署平台，在每套推出的 Oracle 11gR2 RAC 环境中部署 32 个 Oracle11gR2 RAC 备份恢复案例场景脚本。脚本基于网络执行并批量部署；脚本同时支持下载到每一台 Oracle11gR2 RAC 节点主机本地运行。脚本能模拟和评测每一套环境每一个备份恢复场景的恢复结果。

本文中所提及的所有原创程序均提供下载和 md5sum 文件。读者使用这些程序不应出于商业目的，作者对使用这些程序可能带来的一切后果不承担任何法律责任。

目录

正文

1. PXE 概览

1.1 PXE 概览

PXE(Pre-boot Execution Environment)是由 Intel 设计的协议，它可以使裸机通过网络启动，并被安装上操作系统。协议分为推送端和被推送端两端。

PXE 推送端主机是一台协同运行着 dhcpd、tftpd、nfs、dns、vsftpd 和 httpd 的标准 Linux 服务器。

PXE 被推送端程序从网卡的 ROM 中激活。当计算机引导时，BIOS 把 PXE 被推送端程序调入内存执行，并显示出命令菜单。经用户选择后，PXE 被推送端将放置在 PXE 推送端主机的操作系统通过网络下载到本地运行，继而实现 kernel 启动。启动后 PXE 被推送端实际上是主动从 PXE 推送端主机下载安装程序和安装选项文件，完成后续安装步骤。

1.2 IP 地址规划

为了能够把下文叙述清楚，我们设定在 192.168.0.0/24 网段工作。操作系统分别使用 6 系 RHEL 和 5 系 OEL。如果使用其他 6 系 Enterprise Linux 和 5 系 Enterprise Linux（如：CentOS，RHEL 或 OEL，包括它们的 32 位或 64 位发行版）步骤大同小异，本文同样适用。

主机	操作系统	外网 **IP**	内网 **IP**	主机名
PXE 推送端主机	RedHat Enterprise Linux 6.1 x86_64（或其他 6 系 Enterprise Linux 和 5 系 Enterprise Linux）	192.168.0.254	无	server1.example.com
PXE 被推送端主机之 Oracle 11gR2 RAC 环境共享磁盘主机	RedHat Enterprise Linux 6.1（或其他 6 系 Enterprise Linux）	192.168.0.50+x （x 取值范围 1 到 39)	172.31.118.50+x （x 取值范围 1 到 39)	station50+x.example.com （x 取值范围 1 到 39)
PXE 被推送端主机之 Oracle 11gR2 RAC 环境节点主机（第一台）	Oracle Enterprise Linux 5.4（或其他 5 系 Enterprise Linux）	IP：192.168.0.x VIP：192.168.0.200+x SCAN-VIP1：192.168.0.100+x SCAN-VIP2：192.168.0.150+x （x 取值范围 1 到 39)	172.31.118.x （x 取值范围 1 到 39)	stationx.example.com （x 取值范围 1 到 39)
PXE 被推送端主机之 Oracle 11gR2 RAC 环境节点主机（第二台）	同上	IP：192.168.0.1+x VIP：192.168.0.201+x SCAN-VIP3：192.168.0.101+x （x 取值范围 1 到 39)	172.31.118.1+x （x 取值范围 1 到 39)	station1+x.example.com （x 取值范围 1 到 39)

推送时以上 IP 规划通过 PXE 推送端主机上的 dhcpd 配置文件里的 MAC 固定配置来实现。以上 VIP 规划通过 PXE 推送端主机上的 Kickstart 配置文件中“%post 节”编程来实现。而以上三个 SCAN-VIP 规划则通过 PXE 推送端主机的 dns 服务器配置文件配合 PXE 推送端主机上的 Kickstart 配置文件中“%post 节”编程来实现。

2. PXE 推送端主机的搭建

2.1 PXE 推送端主机的搭建

如上所述 PXE 推送端主机是一台协同运行着 dhcpd、tftpd、nfs、dns、vsftpd 和 httpd 的标准 Linux 服务器 。由于本文立足于 Oracle 技术分享，所以并不打算详细介绍这些服务器的具体配置。 为了方便读者搭建 PXE 推送端主机，作者编写了一个一步到位的搭建程序“bcp-install”。

“bcp-install”的功能只有一个：帮助你在 3 分钟之内配置好：一台协同运行着 dhcpd、tftpd、nfs、dns、vsftpd 和 httpd 的标准 Linux 服务器。 此程序实际上是一个 shell 编程的自解压安装包，因此不论操作系统是 32 位和 64 位；也不论是在 6 系 Enterprise Linux 还是在 5 系 Enterprise Linux 上都能运行。

运行该搭建程序前，仅仅要求读者自行安装好一台 Enterprise Linux 主机。 这台主机只需要配备一块网卡，其 IP 和主机名应根据 *1.2 IP* 地址规划里说明的设置好。安装这台主机过程中除了保证“/usr”空间不小于 20G、“/var”空间不小于 15G（如果不分区安装，只要保证“/”不小于 45G）和挑包时装上 ftp 服务器以外没有特殊要求。为了使脚本“bcp-install”能顺利运行成功，需要把安装这台主机操作系统时所用到的整张光盘包含的所有内容，保持完整目录结构直接拷贝至/var/ftp/pub/下。之后运行“bcp-install”：

```
[root@server1 ~]# ./bcp-install
……
SHOULD COPY CDROM/DVDROM of LINUX INSTALLATION into /var/ftp/pub:(y/n)
```

确认把安装这台主机操作系统时所用到的整张光盘包含的所有内容，保持完整目录结构直接拷贝至“/var/ftp/pub/”下，选 y 继续。

至此 PXE 推送端主机的搭建便完成了。如果需要了解“bcp-install”具体做了哪些配置，请打开此机上的 dhcpd、tftpd、nfs、dns、vsftpd 和 httpd 的相应配置文件，辅以 Linux 相关的知识阅读即可。

2.2 下载将要用于推送的安装软件

推送所需的操作系统 iso 镜像和 Oracle 安装软件是不包含在“bcp-install”中的，需要读者自行下载并放置在如下目录（“/usr/sbin/botang-config-push.d/”目录已由“bcp-install”创建）：

所需软件	存放路径
Oracle Enterprise Linux 5.4 iso 镜像文件	/usr/sbin/botang-config-push.d/softwarefiles/Lrhel5/IMAGES/issue.iso
RedHat Enterprise Linux 6.1iso 镜像文件	/usr/sbin/botang-config-push.d/softwarefiles/Lrhel6/IMAGES/issue.iso
linux_11gR2_database_1of2.zip	/usr/sbin/botang-config-push.d/softwarefiles/O11g/DATABASES/linux_11gR2_database_1of2.zip
linux_11gR2_database_2of2.zip	/usr/sbin/botang-config-push.d/softwarefiles/O11g/DATABASES/linux_11gR2_database_2of2.zip
linux_11gR2_grid.zip	/usr/sbin/botang-config-push.d/softwarefiles/O11g/DATABASES/linux_11gR2_grid.zip

2.3 从 PXE 被推送端主机收集 MAC 地址

推送时，*1.2 IP* 地址规划里的 IP 规划是通过 PXE 推送端主机上的 dhcpd 配置文件里的 MAC 固定配置来实现。因此需要编写“/etc/dhcpd.conf”或“/etc/dhcp/dhcpd.conf”配置文件以固定 MAC 和 IP 对应关系。如果读者使用并运行过“bcp-install”，PXE 推送端主机里已安装好一个文本处理工具：“botang-create-dhcpconf”。读者只需要编写“/usr/sbin/workstation.list2”和“/usr/sbin/workstation.list” 两个文件。这两个配置文件前者是 PXE 被推送端主机两块网卡的

MAC 地址列表，后者是 PXE 被推送端主机第一块网卡的 MAC 地址列表。有了这两个文件后，运行 “botang-create-dhcpconf”程序，“/etc/dhcpd.conf”和“/etc/dhcp/dhcpd.conf”配置文件就创建好了。

举例如下：

文件名	内容
/usr/sbin/workstation.list2	3-00:07:E9:5C:44:1F 3-00:16:EC:06:6C:15 4-00:07:E9:0F:74:BC 4-00:16:EC:0B:EA:FE …... 53-00:1A:A0:09:28:62 53-00:E0:50:1A:00:37 …...
/usr/sbin/workstation.list	3-00:07:E9:5C:44:1F 4-00:16:EC:0B:EA:FE …... 53-00:1A:A0:09:28:62 …...

自动生成的：
/etc/dhcp/dhcpd.conf 和/etc/dhcpd.conf
	# Stock gls /etc/dhcpd.conf for RH300 and other classes
	# : dhcpd.conf,v 1.1.2.2 2007/11/19 03:08:34 mcurry Exp $
	ddns-update-style none;
	subnet 192.168.0.0 netmask 255.255.255.0 {
	class "virtual" {
	match if substring (hardware, 1, 3) = 00:16:3e;
	}
	# The following lines are standard all of the time.
	option routers 192.168.0.254;
	option subnet-mask 255.255.255.0;
	option domain-name "example.com";
	option domain-name-servers 192.168.0.254;
	default-lease-time 21600;
	max-lease-time 43200;
	# Assign IPs 61-80 to Xen domUs
	pool {
	allow members of "virtual";
	range 192.168.0.101 192.168.0.189;
	}
	# Assign IPs 1-20 for classroom systems unless
	# static IPs are assigned below
	pool {
	deny members of "virtual";
	range 192.168.0.1 192.168.0.89;
	}
	# The following lines are examples of kickstart
	# directives.
	filename "/var/ftp/pub/workstation.cfg";
	# filename "/kickstart/virt_worstation.cfg;
	# see /root/README.vcracker for details about virt_workstation.cfg
	next-server server1.example.com;
	# The following four lines provide an example of an IP
	# address bound to a specific MAC.
	#
	#
	#
	# Workstation 192.168.0.1-192.168.0.50
	host station3 {
	hardware ethernet 00:07:E9:5C:44:1F;
	fixed-address 192.168.0.3;
	}
	host station4 {
	hardware ethernet 00:16:EC:0B:EA:FE;
	fixed-address 192.168.0.4;
	}
	.....
	host station53 {

```
                hardware ethernet 00:1A:A0:09:28:62;
                fixed-address 192.168.0.53;
}
 ……

}
option space PXE;
class "PXE" {
        match if substring(option vendor-class-identifier, 0, 9) = "PXEClient";
                option vendor-encapsulated-options 01:04:00:00:00:00:ff;
                        option boot-size 0x1;
                        filename "pxelinux.0";
                        option tftp-server-name "server1.example.com";
                        option vendor-class-identifier "PXEClient";
                        vendor-option-space PXE;
}
```

3. PXE 被推送端主机：Oracle 11gR2 RAC 环境共享磁盘主机

现在开始推送 Oracle 11gR2 RAC 环境共享磁盘主机。
在 PXE 推送端主机运行“bcp”:

```
[root@server1 ~]#bcp
……
RAC11grhel5none
RAC11grhel6shareddisknone
Your Choice: RAC11grhel6shareddisknone
……
PUSH CLASSROM MACHINE(c) OR PUSH VIRTUAL MACHINE1(v1) OR PUSH VIRTUAL MACHINE2(v2): (c or v1 or v2)
c
Input is: "c". Are you sure ?(y/n)y
Want rewrite MBR( for before rhel6 should answer "n" ): y
```

PXE 被推送端主机（裸机，硬盘容量不低于 60G），从网卡启动自动被推送。推送过程会在“running post-install script”屏幕菜单处停留较久时间，只需等待，不需干预。推送后，PXE 被推送端主机会自动重启，重启后进入第一个选项，就成为 Oracle 11gR2 RAC 环境共享磁盘主机。

“bcp”程序的主要功能只有一个：在 PXE 推送端主机生成安装选项文件：“/var/ftp/pub/workstation.cfg”。读者若需要了解细节请辅以 Linux 相关的知识阅读。

4. PXE 被推送端主机：Oracle 11gR2 RAC 环境节点主机

现在开始推送 Oracle 11gR2 RAC 环境节点主机。
在 PXE 推送端主机运行 bcp:

```
[root@server1 ~]#bcp
……
RAC11grhel5none
RAC11grhel6shareddisknone
Your Choice:  RAC11grhel5none
……
PUSH CLASSROM MACHINE(c) OR PUSH VIRTUAL MACHINE1(v1) OR PUSH VIRTUAL MACHINE2(v2): (c or v1 or v2)
c
Input is: "c". Are you sure ?(y/n)y
Want rewrite MBR( for before rhel6 should answer "n" ): y
```

PXE 被推送端主机（裸机至少两台，每台硬盘容量不低于 60G），从网卡启动自动被推送。推送过程会在“running post-install script”屏幕菜单处停留较久时间，只需等待，不需干预。推送后，PXE 被推送端主机会自动重启，重启后进入第一个选项，就成为 Oracle 11gR2

RAC 环境节点主机。

“bcp”程序的主要功能只有一个：在 PXE 推送端主机生成安装选项文件：“/var/ftp/pub/workstation.cfg”。读者若需要了解细节请辅以 Linux 相关的知识阅读。

4.1 共享存储

在每套推出的 Oracle 11gR2 RAC 环境中，都包含已自动安装并配置好的三台主机：一台共享磁盘主机和两台节点主机。每套推出的 Oracle 11gR2 RAC 环境中的两台节点主机都已自动挂接上该套环境内部的共享磁盘主机。该共享磁盘已经从“/dev/sdb5”到“/dev/sdb15”分好 11 个等大的分区。读者接下来就可以使用“oracleasm”命令（已自动安装）在 Oracle 11gR2 RAC 环境中的两台节点主机上设定 asm 磁盘。

建议“/dev/sdb5”到“/dev/sdb12”用来创建 normal 冗余度的 asm 磁盘组“+DATA”；“/dev/sdb13”到“/dev/sdb15”用来创建 external 冗余度的 asm 磁盘组“+FRA”。

4.2 分布式虚拟磁带库

在每套推出的 Oracle 11gR2 RAC 环境中的两台节点主机上，各配备一个分布式虚拟磁带库。

```
[root@station34 ~]# cd /stage/mhvtl-1.2/kernel/
[root@station34 kernel]# make
make -C /lib/modules/2.6.18-164.el5/build SUBDIRS=/stage/mhvtl-1.2/kernel modules
make[1]: Entering directory `/usr/src/kernels/2.6.18-164.el5-i686'
  Building modules, stage 2.
  MODPOST
make[1]: Leaving directory `/usr/src/kernels/2.6.18-164.el5-i686'
[root@station34 kernel]# service mhvtl start
vtllibrary process PID is 21614
vtllibrary process PID is 21633
[root@station34 ~]# lsscsi -g
[1:0:0:0]    mediumx STK     L700            0102  -        /dev/sg8
[1:0:1:0]    tape    IBM     ULT3580-TD5     0102  /dev/st0  /dev/sg0
[1:0:2:0]    tape    IBM     ULT3580-TD5     0102  /dev/st1  /dev/sg1
[1:0:3:0]    tape    IBM     ULT3580-TD4     0102  /dev/st2  /dev/sg2
[1:0:4:0]    tape    IBM     ULT3580-TD4     0102  /dev/st3  /dev/sg3
[1:0:8:0]    mediumx STK     L80             0102  -        /dev/sg9
[1:0:9:0]    tape    STK     T10000B         0102  /dev/st4  /dev/sg4
[1:0:10:0]   tape    STK     T10000B         0102  /dev/st5  /dev/sg5
[1:0:11:0]   tape    STK     T10000B         0102  /dev/st6  /dev/sg6
[1:0:12:0]   tape    STK     T10000B         0102  /dev/st7  /dev/sg7
```

osb-10.3.0.3.0_linux32 在/stage 目录下，读者需要自行安装。安装 osb 后，利用/stage 目录下的 osb.sh 脚本，在 Oracle 11gR2 RAC 环境中的两台节点主机上一步到位配置好磁带库：

```
[root@station34 stage]# ./osb.sh
/dev/sg8
/dev/sg9
/dev/sg0
/dev/sg1
/dev/sg2
/dev/sg3
/dev/sg4
/dev/sg5
/dev/sg6
/dev/sg7
Oracle Secure Backup 10.3.0.3.0
Warning: auto-login failed - login token has expired
login: admin
Password:
…...
OSB-CATALOG-MF   write 7 days            keep 14 days
```

```
RMAN-DEFAULT                    content manages reuse
sexample                        content manages reuse
station34                       content manages reuse
```

以 station33、station34 和 station83 这一套环境为例：两台节点主机 station33 和 station34 上的虚拟带库分别设置了“media family”：“station33”和“station34”，还设置了一个基于 RAC 服务的公共“media family”：“sexample”。装好 Oracle 11gR2 RAC 环境后，请在 station33 或 station34 主机上，做 rman 设置，使我们既可以使用磁盘也可以使用分布式虚拟带库做后续备份恢复实验。

```
[oracle@station34 ~]$ rman target /
Recovery Manager: Release 11.2.0.1.0 - Production on Tue Oct 29 10:20:18 2013
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.
connected to target database: ORCL (DBID=1343950367)
RMAN> show all;
using target database control file instead of recovery catalog
RMAN configuration parameters for database with db_unique_name ORCL are:
CONFIGURE RETENTION POLICY TO RECOVERY WINDOW OF 1 DAYS;
CONFIGURE BACKUP OPTIMIZATION ON;
CONFIGURE DEFAULT DEVICE TYPE TO 'SBT_TAPE';
CONFIGURE CONTROLFILE AUTOBACKUP ON;
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE DISK TO '%F'; # default
CONFIGURE CONTROLFILE AUTOBACKUP FORMAT FOR DEVICE TYPE SBT_TAPE TO '%F'; # default
CONFIGURE DEVICE TYPE DISK PARALLELISM 8 BACKUP TYPE TO BACKUPSET;
CONFIGURE DEVICE TYPE 'SBT_TAPE' PARALLELISM 8 BACKUP TYPE TO BACKUPSET;
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE DATAFILE BACKUP COPIES FOR DEVICE TYPE SBT_TAPE TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE DISK TO 1; # default
CONFIGURE ARCHIVELOG BACKUP COPIES FOR DEVICE TYPE SBT_TAPE TO 1; # default
CONFIGURE CHANNEL 1 DEVICE TYPE 'SBT_TAPE' PARMS  'ENV=(OB_MEDIA_FAMILY=station33)'   CONNECT
'sys/oracle_4U@rdbq1';
CONFIGURE CHANNEL 2 DEVICE TYPE 'SBT_TAPE' PARMS  'ENV=(OB_MEDIA_FAMILY=station34)'  CONNECT
'sys/oracle_4U@rdbq2';
CONFIGURE CHANNEL 3 DEVICE TYPE 'SBT_TAPE' PARMS  'ENV=(OB_MEDIA_FAMILY=station33)'  CONNECT
'sys/oracle_4U@rdbq1';
CONFIGURE CHANNEL 4 DEVICE TYPE 'SBT_TAPE' PARMS  'ENV=(OB_MEDIA_FAMILY=station34)'  CONNECT
'sys/oracle_4U@rdbq2';
CONFIGURE CHANNEL 5 DEVICE TYPE 'SBT_TAPE' PARMS  'ENV=(OB_MEDIA_FAMILY=station33)'  CONNECT
'sys/oracle_4U@rdbq1';
CONFIGURE CHANNEL 6 DEVICE TYPE 'SBT_TAPE' PARMS  'ENV=(OB_MEDIA_FAMILY=station34)'  CONNECT
'sys/oracle_4U@rdbq2';
CONFIGURE CHANNEL 7 DEVICE TYPE 'SBT_TAPE' PARMS  'ENV=(OB_MEDIA_FAMILY=station33)'  CONNECT
'sys/oracle_4U@rdbq1';
CONFIGURE CHANNEL 8 DEVICE TYPE 'SBT_TAPE' PARMS  'ENV=(OB_MEDIA_FAMILY=station34)'  CONNECT
'sys/oracle_4U@rdbq2';
CONFIGURE MAXSETSIZE TO UNLIMITED; # default
CONFIGURE ENCRYPTION FOR DATABASE OFF; # default
CONFIGURE ENCRYPTION ALGORITHM 'AES128'; # default
CONFIGURE COMPRESSION ALGORITHM 'BASIC' AS OF RELEASE 'DEFAULT' OPTIMIZE FOR LOAD TRUE ; # default
CONFIGURE ARCHIVELOG DELETION POLICY TO NONE; # default
CONFIGURE SNAPSHOT CONTROLFILE NAME TO '/u01/app/oracle/product/11.2.0/dbhome_1/dbs/snapcf_orcl.f'; # default
```

4.3 时间同步

每套推出的 Oracle 11gR2 RAC 环境中的两台节点主机都已自动时间同步到该套环境内部的共享磁盘主机上的 ntpd 时间服务器。

```
[root@station34 stage]# ntpq
ntpq> peer
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*station83.examp .INIT.          16 u    - 1024    0    0.000    0.000   0.000
ntpq>
```

4.4 grid 用户和 oracle 用户

在每套推出的 Oracle 11gR2 RAC 环境中的两台节点主机上，用户已经被创建好了：“grid”用于安装网格基础架构，“oracle”用于安装数据库。

“grid”用户的环境变量自动配置好了：

```
# .bash_profile
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi
# User specific environment and startup programs
PATH=$PATH:$HOME/bin
export PATH
# export ORACLE_BASE=/u01/app/grid
# export ORACLE_HOME=/u01/app/11.2.0/grid
# export ORACLE_SID=+ASM2
# export TNS_ADMIN=$ORACLE_HOME/network/admin
export NLS_LANG=american_america.AL32UTF8
export ORACLE_TERM=xterm
export EDITOR=vi
export PATH=$ORACLE_HOME/bin:$PATH
export LANG=en_US
```

“oracle”用户的环境变量也自动配置好了：

```
# .bash_profile
# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi
# User specific environment and startup programs
PATH=$PATH:$HOME/bin
export PATH
# export ORACLE_BASE=/u01/app/oracle
# export ORACLE_HOME=$ORACLE_BASE/product/11.2.0/dbhome_1
# export ORACLE_SID=RDBQ2
# export ORACLE_UNQNAME=RDBQ
# export TNS_ADMIN=$ORACLE_HOME/network/admin
export NLS_LANG=american_america.AL32UTF8
export ORACLE_TERM=xterm
export EDITOR=vi
export PATH=$ORACLE_HOME/bin:$PATH
export LANG=en_US
```

这里有个基本安装常识：如果在两个用户、两套“ORACLE_HOME”的环境下安装，安装过程中为避免 listener 注册出现故障，应该先注释掉“ORACLE_BASE”、“ORACLE_HOME”和“TNS_ADMIN”等参数。安装后再释放注释。

4.5 ssh 等价性脚本

每套推出的 Oracle 11gR2 RAC 环境中的两台节点主机上都已创建好了 ssh 等价性脚本：/usr/bin/rac-sshd.sh。安装开始前，在第一台节点上需要用 grid 身份和 oracle 身份各执行一次该脚本：

```
[oracle@station33 stage]$ cat  /usr/bin/rac-sshd.sh
#!/bin/bash
# ssh-keygen, 192.168.0.33->, oracle@192.168.0.33->
echo "KEY GEN....."
sleep 1
echo
echo
```

```
ssh 192.168.0.33 ssh-keygen -t dsa
ssh 192.168.0.34 ssh-keygen -t dsa
echo
echo
# selfconnect, station33->, oracle@station33->
echo "SELF CONNECT....."
sleep 1
echo
echo
ssh station33 cat /home/$USER/.ssh/id_dsa.pub >> /home/$USER/.ssh/authorized_keys
echo
echo
# conterpartconnet, station33.example.com-> , oracle@station33.example.com->
echo "CONTERPART CONNECT....."
sleep 1
echo
echo
scp station34.example.com:/home/$USER/.ssh/id_dsa.pub /tmp/conterpartconnectoracle
ssh station33.example.com cat /tmp/conterpartconnectoracle >> /home/$USER/.ssh/authorized_keys
echo
echo
# TEST
echo "TEST....."
sleep 1
echo
echo
ssh station33 exit
ssh station34 exit
ssh station33.example.com exit
ssh station34.example.com exit
echo
echo
# 192.168.0.34->, oracle@192.168.0.34->
echo "SCP TO CONTERPART ....."
sleep 1
echo
echo
scp /home/$USER/.ssh/known_hosts $USER@192.168.0.34:/home/$USER/.ssh/known_hosts
scp /home/$USER/.ssh/authorized_keys $USER@192.168.0.34:/home/$USER/.ssh/authorized_keys
echo
echo
[oracle@station33 stage]$ rac-sshd.sh
```

4.6 其他

每套推出的 Oracle 11gR2 RAC 环境中的两台节点主机都已经做好所有操作系统层面的配置：包括 vip、scan-vip、ntp，各种 rpm 包和操作系统参数等。所有安装软件都已经下载到“/stage”目录下。推送出的所有主机上的 root 用户和其他操作系统用户的密码都是“ oracle”。

4.7 安装

使用以上环境安装各套 Oracle 11gR2 RAC 环境，系统自检是不会报告任何错误的。全局数据库名和实例名前缀应根据自动生成的.bash_profile 文件的规定来建库。比如：station33，station34 和 station83 这一套的 SID 就是 RDBQ1 和 RDBQ2。注意所有 PXE 被推送端主机上生成的脚本和配置文件在推送过程中都是自动根据 IP 规划变量调整其内容中出现的主机名或 SID 名的。

如果要继续后面的备份恢复实验，sys、system、sysman 和 dbsnmp 的密码都必须设置成：“oracle_4U”。必需安装示例方案。

具体安装过程都是标准流程，这里略去。如有需要请参考上海 Oracle 用户组其他成员的相关共享文档。

```
[root@station33 ~]# crs_stat -t
```

```
Name           Type           Target    State     Host
-----------------------------------------------------------
ora.DATA.dg    ora....up.type ONLINE    ONLINE    station33
ora.FRA.dg     ora....up.type ONLINE    ONLINE    station33
ora....ER.lsnr ora....er.type ONLINE    ONLINE    station33
ora....N1.lsnr ora....er.type ONLINE    ONLINE    station34
ora....N2.lsnr ora....er.type ONLINE    ONLINE    station33
ora....N3.lsnr ora....er.type ONLINE    ONLINE    station33
ora.asm        ora.asm.type   ONLINE    ONLINE    station33
ora.eons       ora.eons.type  ONLINE    ONLINE    station33
ora.gsd        ora.gsd.type   ONLINE    ONLINE    station33
ora....network ora....rk.type ONLINE    ONLINE    station33
ora.oc4j       ora.oc4j.type  ONLINE    ONLINE    station33
ora.ons        ora.ons.type   ONLINE    ONLINE    station33
ora.rdbq.db    ora....se.type ONLINE    ONLINE    station33
ora....ry.acfs ora....fs.type ONLINE    ONLINE    station33
ora.scan1.vip  ora....ip.type ONLINE    ONLINE    station34
ora.scan2.vip  ora....ip.type ONLINE    ONLINE    station33
ora.scan3.vip  ora....ip.type ONLINE    ONLINE    station33
ora....SM1.asm application    ONLINE    ONLINE    station33
ora....33.lsnr application    ONLINE    ONLINE    station33
ora....n33.gsd application    ONLINE    ONLINE    station33
ora....n33.ons application    ONLINE    ONLINE    station33
ora....n33.vip ora....t1.type ONLINE    ONLINE    station33
ora....SM2.asm application    ONLINE    ONLINE    station34
ora....34.lsnr application    ONLINE    ONLINE    station34
ora....n34.gsd application    ONLINE    ONLINE    station34
ora....n34.ons application    ONLINE    ONLINE    station34
ora....n34.vip ora....t1.type ONLINE    ONLINE    station34
```

5. 用 Shell 脚本在推出的节点上批量部署 32 个 Oracle11gR2 RAC 备份恢复案例场景

5.1 备份恢复案例场景模拟脚本工作原理

基本工作原理是：读者按照以上推送设置的要求安装好成套 Oracle11gR2 RAC 环境后，复用以上的 PXE 推送端主机作为部署平台，在每套推出的 Oracle 11gR2 RAC 环境中部署 32 个 Oracle11gR2 RAC 备份恢复案例场景脚本。脚本基于网络执行并批量部署；脚本同时支持下载到每一台 Oracle11gR2 RAC 节点主机本地运行。脚本能模拟和评测每一套环境每一个备份恢复场景的恢复结果。

32 个备份恢复场景模拟脚本调用一个公共子程序库“bclcustom-subprogram”。在该公共子程序库里面编写了各个备份恢复场景标准步骤。32 个备份恢复场景模拟脚本实际上是根据场景模拟需要组合这些标准步骤来完成相应的场景模拟的。

公共子程序库“bclcustom-subprogram”如下：

```
####
#### 0 Subprograms Begin
###  0.1 Normal Structure Begin
###

# sub_getting has $1 $2
sub_getting() {
 echo " "
 echo "------------------------------------------- LAB $1 -------------------------------------------"
 echo "            $2"
 echo "------------------------------------------- LAB $1 -------------------------------------------"
 echo " "
 echo " "
 echo "******************"
 echo "你的主机信息如下： "
 echo "******************"
 ifconfig eth0 | head -n2 | tail -n1 | cut -f 2 -d : | cut -f 1 -d ' '
```

```
 echo " "
 hostname
 echo " "
 uname -a
 echo " "
 echo " "
}

sub_detecting() {
 /u01/app/11.2.0/grid/bin/crs_stat -t
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "sqlplus /nolog" <<!
  conn / as sysdba
  set echo off
  set feedback off
  @/home/oracle/ptable.sql;
  quit
!
 fi
 echo " "
}

sub_revealing() {
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then

  su - oracle -c "sqlplus /nolog" <<!
  conn / as sysdba
  set echo off
  set feedback off
  @/home/oracle/ttable.sql;
  quit
!
 fi
 echo " "
}

sub_resultscoring() {
 /u01/app/11.2.0/grid/bin/crs_stat -t
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "sqlplus /nolog" <<!
  conn  sys/oracle_4U@$v_dbname as sysdba
  set echo off
  set feedback off
  @/home/oracle/rtable.sql;
  quit
!
 fi
 echo " "
}

sub_clearing() {
 rm -f /home/oracle/ptable.sql
 rm -f /home/oracle/ctable.sql
 rm -f /home/oracle/ttable.sql
 rm -f /home/oracle/rtable.sql
 rm -f /usr/bin/bclcustom.sh
 rm -f /home/oracle/asm
 rm -f /home/oracle/bclcustom-subprogram
}

sub_clearing_excludertable() {
 rm -f /home/oracle/ptable.sql
 rm -f /home/oracle/ctable.sql
```

```
 rm -f /home/oracle/ttable.sql
 rm -f /usr/bin/bclcustom.sh
 rm -f /home/oracle/asm
 rm -f /home/oracle/bclcustom-subprogram
}

###
###   0.1 Normal Structure End
###   0.2 Database Operation Begin
##    0.2.1 Database SRVCTl Operation Begin
##

sub_shutdowning_normal() {
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "/u01/app/oracle/product/11.2.0/dbhome_1/bin/srvctl stop database -d $v_dbname"
 fi
 echo " "
}

##
##    0.2.1 Database SRVCTl Operation End
##    0.2.2 Database ASM Operation Begin
##

# sub_destroying has $1
sub_destroying() {

 # $1 : system*
 #   #      undotbs1*
 #     #      undotbs2*
 #       #      sysaux*
 #         #      example*
 #           #      users*
 #             #      group*
 #               #      current*
 #
 su - grid -c "export ORACLE_SID=+ASM1 ; export ORACLE_HOME=/u01/app/11.2.0/grid ; asmcmd "<<! > /home/oracle/asm
  find / $1
  exit
!
  sleep 15
  sync
  v_path1=`cut -f 2 -d '+' /home/oracle/asm  | tr -d [:blank:]`
  for i in $v_path1
  do
   v_path2="+"$i
   su - grid -c "export ORACLE_SID=+ASM1 ; export ORACLE_HOME=/u01/app/11.2.0/grid ; asmcmd "<<!
   rm $v_path2
   exit
!
  done
 echo " "
}

##
##    0.2.2 Database ASM Operation End
##    0.2.3 Database SQL Operation Begin
##

sub_startuping_node1mount() {
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "sqlplus /nolog" <<!
  conn / as sysdba
  startup mount exclusive
  quit
!
```

```
 fi
 echo " "
}

sub_creating() {
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "sqlplus /nolog" <<!
  conn sys/oracle_4U@$v_dbname as sysdba
  set echo off
  set feedback off
  @/home/oracle/ctable.sql;
  quit
!
 fi
 echo " "
}

sub_scripting_controlfile_nocatalog() {
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 echo "Next moment control script is going to be create ....."
 echo "Display the spid number of session here: "
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "sqlplus /nolog" <<!
  conn sys/oracle_4U@$v_dbname as sysdba
  select instance_number from v\$instance;
  select spid from v\$process p , v\$session  s , v\$mystat  m  where p.addr=s.paddr and m.sid=s.sid  and rownum=1;
  alter database backup controlfile to trace;
  quit
!
 fi
 echo ""
}

sub_offlining_users_immediate() {
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "sqlplus /nolog" <<!
  conn  sys/oracle_4U@$v_dbname as sysdba
  set echo off
  set feedback off
  select instance_number from v\$instance;
  alter tablespace users offline immediate;
  quit
!
 fi
 echo " "
}

sub_offlining_tbsocp05_test_immediate() {
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "sqlplus /nolog" <<!
  conn  sys/oracle_4U@$v_dbname as sysdba
  set echo off
  set feedback off
  select instance_number from v\$instance;
  alter tablespace tbsocp05_test offline immediate;
  quit
!
 fi
 echo " "
}

sub_shutdowning_abort() {
```

```
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "/u01/app/oracle/product/11.2.0/dbhome_1/bin/srvctl stop database -d $v_dbname -o abort"
 fi
}

sub_altering_default_tablespace() {
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "sqlplus /nolog" <<!
  conn / as sysdba
  alter database default tablespace example;
  quit
!
 fi
 echo ""
}

sub_dropping_users() {
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  echo "***FROM THEN ON: DROP***"
  su - oracle -c "sqlplus /nolog" <<!
  conn / as sysdba
  drop user sh cascade;
  drop user oe cascade;
  drop tablespace users including contents and datafiles;
  quit
!
 fi
 echo " "
}

##
##    0.2.3 Database SQL Operation End
##    0.2.4 Database RMAN Operation Begin
##
# sub_backingup_controlfile_nocatalog has $1
sub_backingup_controlfile_nocatalog() {
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "rman target sys/oracle_4U@$v_dbname" <<!
  backup tag "$1" current controlfile;
  quit
!
 fi
 echo ""
}

sub_deleting_backup_controlfile_nocatalog() {
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "rman target sys/oracle_4U@$v_dbname" <<!
  delete noprompt backup of controlfile;
  delete noprompt copy of controlfile;
!
 fi
 echo ""
}

sub_resync_catalog() {
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 v_ip=$(ifconfig | grep  '192\.168\.0' | head -n 1| cut -d . -f 4| cut -d ' ' -f 1)
 grep 'LABS ADD' /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora
```

```
 if [ $? == '1' ]
 then
  echo " " >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora
  echo "rcat =" >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora
  echo "  (DESCRIPTION =" >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora
  echo "    (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.0.90)(PORT = 1521))" >>
/u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora
  echo "    (CONNECT_DATA =" >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora
  echo "      (SERVER = DEDICATED)" >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora
  echo "      (SERVICE_NAME = rcat.example.com)" >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora
  echo "     )" >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora
  echo "   )" >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora
  echo " " >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora
  echo '#LABS ADD' >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora
 fi
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  echo "***Connecting Catalog***"
  su - oracle -c "rman target sys/oracle_4U@$v_dbname catalog u90/oracle_4U@rcat" <<!
  resync catalog;
!
 fi
 echo " "
}

##
##    0.2.4 Database RMAN Operation End
##    0.2.5 Database SQL/RMAN Combined Operation Begin
##
# sub_onlining_users_nocatalog has $1
sub_onlining_users_nocatalog() {
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "sqlplus /nolog" <<!
  conn  sys/oracle_4U@$v_dbname as sysdba
  select instance_number from v\$instance;
  alter tablespace users online;
  quit
!
 fi
 echo " "
 # rman
 echo "***AFTER TABLESPACE ONLINE***"
 echo "***BACKUP AGAIN***"
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "rman target sys/oracle_4U@$v_dbname" <<!
  backup tag "$1" tablespace users;
!
 fi
 echo " "
}

# sub_offlining_users_nocatalog has $1
sub_offlining_users_nocatalog() {
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "sqlplus /nolog" <<!
  conn  sys/oracle_4U@$v_dbname as sysdba
  select instance_number from v\$instance;
  alter tablespace users offline;
  quit
!
 fi
 echo " "
 # rman
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
```

```
 then
  su - oracle -c "rman target sys/oracle_4U@$v_dbname"<<!
  backup tag "$1" tablespace users;
!
 fi
 echo " "
}

# sub_readingwrite_users_nocatalog has $1
sub_readingwrite_users_nocatalog() {
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "sqlplus /nolog" <<!
  conn  sys/oracle_4U@$v_dbname as sysdba
  select instance_number from v\$instance;
  alter tablespace users read write;
  quit
!
 fi
 echo " "
 # rman
 echo "***AFTER TABLESPACE REWRITE***"
 echo "***BACKUP AGAIN***"
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "rman target sys/oracle_4U@$v_dbname" <<!
  backup tag "$1" tablespace users;
!
 fi
 echo " "
}

# sub_readingonly_users_nocatalog has $1
sub_readingonly_users_nocatalog() {
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "sqlplus /nolog" <<!
  conn  sys/oracle_4U@$v_dbname as sysdba
  select instance_number from v\$instance;
  alter tablespace users read only;
  quit
!
 fi
 echo " "
 # rman
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - oracle -c "rman target sys/oracle_4U@$v_dbname" <<!
  backup tag "$1" tablespace users;
!
 fi
 echo " "
}

sub_deleting_early_online_backup_users() {
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  echo "***DELETE THE EARLY ONLINE BACKUP OF USERS***"
  echo "***DELETE THE EARLY READ WRITE BACKUP OF USERS***"
  su - oracle -c "rman target sys/oracle_4U@$v_dbname" <<!
  delete noprompt backup of tablespace users;
  delete noprompt copy of tablespace users;
!
  sub_deleting_backup_controlfile_nocatalog
  fi
 echo " "
```

```
}

##
##    0.2.5 Database SQL/RMAN Combined Operation End
###   0.2 Database Operation End
####  0 Subprograms End
####
```

从以上脚本可以看出，许多实验需要 catalog 恢复目录。在脚本中规定恢复目录存在于一台 IP 为 192.168.0.90 的主机上，服务名为“rcat.example.com”，监听端口为：“1521”。恢复目录用户名为“u90”，密码为“oracle_4U”。请事先准备好该设备。

5.2 在 PXE 推送端主机上安装备份恢复案例场景模拟脚本

为了方便读者，作者编写了一个一步到位的安装程序“bcl-install”。此程序在 PXE 推送端主机上安装。

“bcl-install”包含的主要内容就是上面的子程序。“bcl-install”程序实际上是一个 shell 编程的自解压安装包，因此不论操作系统是 32 位和 64 位；也不论是在 6 系 Enterprise Linux 还是在 5 系 Enterprise Linux 上都能运行。

```
[root@server1 ~]# ./bcl-install
```

5.3 在 Oracle 11gR2 RAC 环境节点主机部署备份恢复案例场景模拟脚本

现在准备开始在 Oracle 11gR2 RAC 环境节点主机部署备份恢复案例场景模拟脚本。

5.3.1 在 PXE 推送端主机上准备 IP 地址列表

为了指定进行备份恢复案例场景模拟的 Oracle 11gR2 RAC 环境节点主机列表， 请编写“/tmp/botang-config-xshare/course/general/v_botang_all_ipsshd_open”文件，把每套要被模拟备份恢复场景的 Oracle 11gR2 RAC 环境节点主机的 IP 地址以空格隔开写成一行，以 192.168.0.33 和 192.168.0.34 举例如下：

```
[root@server1 ~]# cd /tmp/botang-config-xshare/course/general
[root@server1 general]# cat v_botang_all_ipsshd_open
192.168.0.33 192.168.0.34
```

5.3.2 在 PXE 推送端主机上初始化实验脚本

```
[root@server1 ~]# bcl --RACGRID11g13 newlabs
……
Password Number is: 5.
RAC FITABLE
The course for you to decide: ?
general

*****************************************************************************************************
一般注意事项：
 1.以下大多数实验要求数据库处于归档模式，一些实验要求有全库备份，请一定注意 users 表空间要有单独的备份通道（使它自己包含在它自己的备份集中）。
 2.把压缩包中各个实验脚本中的*.sql 和 bclcustom-subprogram 放在/home/oracle/下面， 以 root 身份运行 Cbclcustom.sh(可以放在任何地方)，即可实施破坏并作实验。
 3.sys 的密码一定要叫做 oracle_4U。
 4.实验环境的 grid 和 db_home1 一定要按照课程环境安装。
 5.唯独实验 6b 和实验 15 需要在执行完成 Cbclcustom.sh 之后运行各自文件夹下面的 begin-realpost。作进一步破坏，请打开 begin-realpost 阅读一下就明白了。
 6.11g 没有实验 2。
```

```
**********************************************************************************************
RACGRID11g13.ori/
RACGRID11g13.ori/7a_当前控制文件和数据文件完好_日志文件全部损坏_正常关机不完全恢复_数据不丢_不需备份/
RACGRID11g13.ori/7a_当前控制文件和数据文件完好_日志文件全部损坏_正常关机不完全恢复_数据不丢_不需备份/begin-real
RACGRID11g13.ori/7a_当前控制文件和数据文件完好_日志文件全部损坏_正常关机不完全恢复_数据不丢_不需备份/bclcustom-subprogram
……
```

在 PXE 推送端主机上初始化实验脚本的目的是：解压缩脚本压缩包“/usr/sbin/botang-config-lesson.d/CUSTOM/RACGRID11g13.ori.tgz”到“/usr/sbin/botang-config-lesson.d/CUSTOM/RACGRID11g13.ori”，供调用。

“/usr/sbin/botang-config-lesson.d/CUSTOM/RACGRID11g13.ori”里每一个文件夹，就是一个备份恢复场景模拟脚本。举例如下：

```
[root@station90 ~]# cd /usr/sbin/botang-config-lesson.d/CUSTOM/RACGRID11g13.ori/14_删除表空间不完全恢复
[root@station90 14_删除表空间不完全恢复]# ls
bclcustom-subprogram  begin-real    ctable.sql  Pbclcustom.sh  Rbclcustom.sh  ttable.sql       vsftpdrestart-prepare
begin-prepare         Cbclcustom.sh  end         ptable.sql     rtable.sql     vsftpdrestart-end  vsftpdrestart-real
[root@station90 14_删除表空间不完全恢复]#
```

“bcl”程序在批量部署时，自动完成以下四个阶段工作：

第一阶段：拷贝该场景下“*.sql 和“bclcustom-subprogram”到 Oracle 11gR2 RAC 环境节点主机的“/home/oracle”下。

第二阶段：通过 ssh 调用该场景下的“ Pbclcustom.sh”在 Oracle 11gR2 RAC 环境节点主机运行，其标准输出收集起来形成 ftp 页面：“ftp://192.168.0.245/pub/bclresult-xx/prepare”，供查阅。“ Pbclcustom.sh”会自动判断奇数/偶数节点，会在奇数执行而绕过 偶数节点。“ Pbclcustom.sh”的主要任务是判断 Oracle 11gR2 RAC 环境节点主机的健康状态。

第三阶段：通过 ssh 调用该场景下的“ Cbclcustom.sh”在 Oracle 11gR2 RAC 环境节点主机运行，其标准输出收集起来形成 ftp 页面：“ftp://192.168.0.245/pub/bclresult-xx/real”，供查阅。“ Cbclcustom.sh”会自动判断奇数/偶数节点，会在奇数执行而绕过 偶数节点。“ Cbclcustom.sh”的主要任务是在 Oracle 11gR2 RAC 环境节点主机搞破坏，模拟场景。

“ Cbclcustom.sh”程序会调用“ctable.sql”脚本，后者会在每套 Oracle 11gR2 RAC 环境节点主机该场景中建立一个表名为“t”打头加上随机数的表。这样保证了每套 Oracle 11gR2 RAC 环境节点在本场景中表名的唯一性。只有现场恢复了，才能找到该表。使得这套程序能适用考试和面试。

“ctable.sql”脚本会伴随着日志切换插入该表“1、2、3、4 和 5”共 5 行数值。实验最后的“Rbclcustom.sh”脚本会记住每套 Oracle 11gR2 RAC 环境节点主机该场景中建立的那个表，并查询其数值，评测该备份恢复场景的恢复结果。如该场景是完全恢复能找到“1、2、3、4 和 5”5 行数值；如场景是不完全恢复而且无法找全数值，由于“4 和 5”两个数值对应的 redo log entry 是在丢失或不可用的日志上，只能找到“ 1、2 和 3”（在 rac 环境下，由于日志 thread 的交替，可能数值 3 也找不到）。每个场景的“ctable.sql”都由公共子程序库“bclcustom-subprogram”中的子程序 sub_creating
调用，但是每个场景的“ctable.sql”都不一样，即在数据库中做的场景特定操作。

对于编号为“6b”的实验，系统还需自动多走一步：在每套 Oracle 11gR2 RAC 环境节点主机上删除特定日志（每套系统都会判断是哪一个日志），其标准输出收集起来替换盖掉日志序列号形成 ftp 页面：“ftp://192.168.0.245/pub/bclresult-xx/real-post”。同时暂时锁住“ftp://192.168.0.245/pub/bclresult-xx/real”。

对于编号为“15”的实验， 系统还需自动多走两步：在每套 Oracle 11gR2 RAC 环境节点主机上删除特定日志（每套系统都会判断是哪一个日志），并将该场景恢复掉，制造出一个新 incarnation--“3 号化身”，以便让用户之后穿越 incarnation 恢复到“2 号化身”。其标准输出收集起来形成 ftp 页面：“ftp://192.168.0.245/pub/bclresult-xx/real-post”。

第四阶段：通过 ssh 调用该场景下的“ Rbclcustom.sh”在 Oracle 11gR2 RAC 环境节点主机运行，其标准输出收集起来形成 ftp 页面：“ftp://192.168.0.245/pub/bclresult-xx/end”，供查阅。“ Rbclcustom.sh”会自动判断奇数/偶数节点，会在奇数执行而绕过 偶数节点。如前所述，“

Rbclcustom.sh”的主要任务是在 Oracle 11gR2 RAC 环境节点主机 评测该备份恢复场景的恢复结果。

特殊的情况：对于每一个场景，要做实验的用户也可自行把压缩包中对应的实验脚本中的“*.sql”和“bclcustom-subprogram”放在 Oracle 11gR2 RAC 环境节点主机“/home/oracle/”下面，自行以 root 身份依次运行“Pbclcustom.sh”自检环境-->“Cbclcustom.sh”实施破坏并作实验-->最后运行“Rbclcustom.sh”收集实验结果。所不同的是：标准输出直接在屏幕上，从“prepare”到“real”到“end”的整个过程表名都是“edu234”（而不是随机表名）。

6. Oracle11gR2 RAC 环境“dd”和“rever”，在后续特定实验前通过“rever”初始化环境

在开始实验之前，显然要求每套 Oracle 11gR2 RAC 环境都要根据推送时的标准安装好。并都已经开启归档、已经做好 rman 全库备份（无论磁带机或磁盘备份）。为了方便标识，建议在每套 Oracle 11gR2 RAC 环境中都建一张表：

```
SQL>conn / as sysdba
SQL>create table sys.tname(a varchar2(50));
SQL>insert into sys.tname values ('用户 ID');
```

如果读者需要连续做实验，在后续特定实验前，为了避免实验间连续恢复产生的过度复杂性，建议读者通过冷备份 rever 初始化环境。

每套 Oracle 11gR2 RAC 环境，在做好以上所有工作并准备做第一个实验之前，请手工冷备份整套环境。以“oracle”用户在每套 Oracle 11gR2 RAC 环境第一台节点主机上运行：“/home/oracle/copy.sh”能达到手工冷备份整套环境的目的。以后可以在任何时候，通过以“oracle”用户运行“/home/oracle/rever.sh”初始化环境。如果该套 Oracle 11gR2 RAC 环境使用磁带机做备份，请以“root”身份运行“copytape-root.sh”备份虚拟磁带库。当以“oracle”用户运行“/home/oracle/rever.sh”初始化环境后，都要再以“root”身份运行“revertape-root.sh”初始化虚拟带库。并重新注册 catalog。

通过在 PXE 推送端主机上运行：

```
[root@server1 ~]# bcl --RACGRID11g13 0
三个问题都选择 y
```

以上命令把脚本“copy.sh”和“rever.sh”，直接推到每套 Oracle 11gR2 RAC 环境第一台节点主机“/home/oracle”之下。同时脚本“copy.sh”和“rever.sh”在 PXE 推送端主机 ftp 站点：“ftp://192.168.0.254/pub/bclresult-0/prepare”也可下载到。“copytape-root.sh”和“ revertape-root.sh”虽然在 bcl 程序中不提供，但是它们的内容列在脚本“copy.sh”和“rever.sh”之后。有需要的，将其编写在每套 Oracle 11gR2 RAC 环境所有节点主机上（任何路径），以“root”身份并运行。

“copy.sh”内容如下：

```
#!/bin/sh
###
### 0 Subprogram Section Begin
###

v_start_time=$(date +%"s")
sub_confirm() {
        sub_answer() {
          unset SUBANS
                sub_sub_answer(){
                SUBANS=`echo $SUBANS |tr -d "[:blank:]"`
                if [ -z $SUBANS ]
                then
                SUBANS="NULL"
```

```
                fi
        }
        #
        # Sub_sub_answer end.
        #
        read -p "$1" SUBANS
        sub_sub_answer # Line15
        # Line15 begin: Deal with $SUNANS:
        until [ $SUBANS ==  "y"  -o  $SUBANS ==  "n" ]
        do
                echo "This question should be answered either with \"y\" or \"n\"."
                read -p "$1" SUBANS
                sub_sub_answer
        done
        # Line15 end.
    }
    #
    # Sub_answer end.
    #
    unset SUBCON1
    unset SUBCON2
    SUBCON1=0
    SUBCON2=1
    until [ $(echo $SUBCON1|tr -d "[:blank:]")  == $(echo $SUBCON2|tr -d "[:blank:]") ]
    do
        read -p "$1" SUBCON1
        sub_answer "Your input is: \"$SUBCON1\". Are you sure ?(y/n) "
        if [ -z $(echo $SUBCON1|tr -d "[:blank:]") ]
        then
         SUBCON1="NULL"
        fi
        if [ $SUBANS == "y" ]
        then
        SUBCON2=$SUBCON1
        fi
    done
}
v_dbname=`cat /home/oracle/dbname 2>/dev/null | tr -d [:blank:]`
v_first_node=`cat /home/oracle/nodeinfo 2>/dev/null | head -n 1`
v_second_node=`cat /home/oracle/nodeinfo 2>/dev/null | tail -n 1`

###
### 0 Subprogram Section End
### 1 Main Section Begin
###

# 1 Decision Root or Oracle Begin
if [ $(id -u) == 0 ]
then
 echo "Please login as oracle NOT root."
 exit
fi
# 1 Decision Root or Oracle End
# 2 Stop DB Begin
if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
then
 if [ -z $v_dbname ]
 then
  sub_confirm "Please specify the database name: ? "
  v_dbname=$SUBCON1
  echo $v_dbname>/home/oracle/dbname
 fi
 if [ -z $v_first_node ]
 then
  sub_confirm "Please specify the first node name: ? "
  v_first_node=$SUBCON1
 fi
 if [ -z $v_second_node ]
 then
```

```
 sub_confirm "Please specify the second node name: ? "
 v_second_node=$SUBCON1
fi
srvctl stop instance -d $v_dbname -i ${v_dbname}1
sudo /u01/app/11.2.0/grid/bin/crs_stat -t
srvctl stop instance -d $v_dbname -i ${v_dbname}2
sudo /u01/app/11.2.0/grid/bin/crs_stat -t
sudo /u01/app/11.2.0/grid/bin/crsctl stop has
ssh $v_second_node "sudo /u01/app/11.2.0/grid/bin/crsctl stop has"
echo "Please wait ....."
sleep 60
fi
/usr/bin/stopdb &>/dev/null
# 2 Stop DB End
# 3 Man Directory Begin
if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
then
 if [ ! -d /u01/data/backup ]
 then
  sub_confirm "Please input the full path of directory to hold the 11 rawdevices backup of RAC, need about 18G ( oracle user should have !!!WRITE PERMISSION!!! in it): ? "
 else
  SUBCON1=/u01/data
 fi
  rm -rf  $SUBCON1/backup 2>/dev/null
  mkdir -p $SUBCON1/backup 2>/dev/null
  chown oracle:oinstall $SUBCON1/backup
fi
#rm -rf /u01/app/oracle/man_recovery_area/orcl/backup/
mkdir -p /u01/app/oracle/man_recovery_area/orcl 2>/dev/null
# 3 Man Directory End
# 4 Backup Important Begin
if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
then
 for i in `ls /dev/oracleasm/disks`
 do
  dd if=/dev/oracleasm/disks/$i of=$SUBCON1/backup/$i.img bs=4M &
 done
 wait
fi
# 4 Backup Important End
# 5 Backup Miscellaneous Begin
# 5 Backup Miscellaneous End
# 6 Start DB Begin
if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
then
 sudo /u01/app/11.2.0/grid/bin/crsctl start has
 ssh $v_second_node "sudo /u01/app/11.2.0/grid/bin/crsctl start has"
 v_end_time=$(date +%"s")
 echo "总共花了： $[ $v_end_time - $v_start_time ]秒。再等 90 秒为你启动集群。"
 sleep 90
 sudo /u01/app/11.2.0/grid/bin/crsctl start resource ora.DATA.dg
 sudo /u01/app/11.2.0/grid/bin/crsctl start resource ora.FRA.dg
 srvctl start database -d $v_dbname
 sudo /u01/app/11.2.0/grid/bin/crs_stat -t
fi
# 6 End DB End
```

“rever.sh”内容如下：

```
#!/bin/sh
###
### 0 Subprogram Section Begin
###

v_start_time=$(date +%"s")
sub_confirm() {
        sub_answer() {
```

```
            unset SUBANS
            #
            # Sub_sub_answer begin:
                  sub_sub_answer(){
                  SUBANS=`echo $SUBANS |tr -d "[:blank:]"`
                  if [ -z $SUBANS ]
                  then
                  SUBANS="NULL"
                  fi
            }
            #
            # Sub_sub_answer end.
            #
            read -p "$1" SUBANS
            sub_sub_answer # Line15
            # Line15 begin: Deal with $SUNANS:
            until [ $SUBANS ==  "y"  -o  $SUBANS ==  "n" ]
            do
                  echo "This question should be answered either with \"y\" or \"n\"."
                  read -p "$1" SUBANS
                  sub_sub_answer
            done
            # Line15 end.
      }
      #
      # Sub_answer end.
      #
      unset SUBCON1
      unset SUBCON2
      SUBCON1=0
      SUBCON2=1
      until [ $(echo $SUBCON1|tr -d "[:blank:]")  == $(echo $SUBCON2|tr -d "[:blank:]") ]
      do
            read -p "$1" SUBCON1
            sub_answer "Your input is: \"$SUBCON1\". Are you sure ?(y/n) "
            if [ -z $(echo $SUBCON1|tr -d "[:blank:]") ]
            then
             SUBCON1="NULL"
            fi
            if [ $SUBANS == "y" ]
            then
            SUBCON2=$SUBCON1
            fi
      done
}
v_dbname=`cat /home/oracle/dbname 2>/dev/null | tr -d [:blank:]`
v_first_node=`cat /home/oracle/nodeinfo 2>/dev/null | head -n 1`
v_second_node=`cat /home/oracle/nodeinfo 2>/dev/null | tail -n 1`

###
### 0 Subprogram Section End
### 1 Main Section Begin
###

# 1 Decision Root or Oracle Begin
if [ $(id -u) == 0 ]
then
 echo "Please login as oracle NOT root."
 exit
fi
# 1 Decision Root or Oracle End
# 2 Stop DB Begin
if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
then
 v_dbname=`cat /home/oracle/dbname | tr -d [:blank:]`
 v_first_node=`cat /home/oracle/nodeinfo | head -n 1 | tr -d [:blank:]`
 v_second_node=`cat /home/oracle/nodeinfo | tail -n 1 | tr -d [:blank:]`
 srvctl stop instance -d $v_dbname -i ${v_dbname}1 -o abort
 sudo /u01/app/11.2.0/grid/bin/crs_stat -t
```

```
 srvctl stop instance -d $v_dbname -i ${v_dbname}2 -o abort
 sudo /u01/app/11.2.0/grid/bin/crs_stat -t
 sudo /u01/app/11.2.0/grid/bin/crsctl stop has
 ssh $v_second_node "sudo /u01/app/11.2.0/grid/bin/crsctl stop has"
 echo "Please wait ....."
 sleep 60
fi

emctl stop dbconsole

# 2 Stop DB End
# 3 Delete Begin
# 3 Delete End
# 4 Recovery Begin

if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
then
 if [ ! -d /u01/data/backup ]
 then
  sub_confirm "Please input the full path of directory to hold the 11 rawdevices backup of RAC, need about 18G: ? "
 else
  SUBCON1=/u01/data
 fi
 for i in `ls /dev/oracleasm/disks`
 do
  dd of=/dev/oracleasm/disks/$i if=$SUBCON1/backup/$i.img bs=4M&
 done
 wait
fi
# 4 Recovery End
if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
then
 sudo /u01/app/11.2.0/grid/bin/crsctl start has
 ssh $v_second_node "sudo /u01/app/11.2.0/grid/bin/crsctl start has"
 v_end_time=$(date +%"s")
 echo "总共花了： $[ $v_end_time - $v_start_time ]秒。"
 sleep 90
 sudo /u01/app/11.2.0/grid/bin/crsctl start resource ora.DATA.dg
 sudo /u01/app/11.2.0/grid/bin/crsctl start resource ora.FRA.dg
 srvctl start database -d $v_dbname
 sudo /u01/app/11.2.0/grid/bin/crs_stat -t
fi
```

“copytape-root.sh”内容如下：

```
#!/bin/sh
service observiced stop
service mhvtl stop
rmmod mhvtl

mkdir -p /u01/app/oracle/man_recovery_area/orcl/backup/tape 2>/dev/null
cd /opt/ ; tar -zcvf /u01/app/oracle/man_recovery_area/orcl/backup/tape/mhvtl.tgz  mhvtl
cd /usr/local/oracle/ ; tar -zcvf /u01/app/oracle/man_recovery_area/orcl/backup/tape/backup.tgz  backup
cd /usr/etc/; tar -zcvf /u01/app/oracle/man_recovery_area/orcl/backup/tape/ob.tgz ob
chown -R oracle:oinstall /u01/app/oracle/man_recovery_area/orcl/backup/tape/

service mhvtl start
service observiced start
```

“ revertape-root.sh”内容如下：

```
#!/bin/sh
service observiced stop
service mhvtl stop
rmmod mhvtl
```

```
cd /opt/ ; tar -zxvf /u01/app/oracle/man_recovery_area/orcl/backup/tape/mhvtl.tgz
cd /usr/local/oracle/ ; tar -zxvf /u01/app/oracle/man_recovery_area/orcl/backup/tape/backup.tgz
cd /usr/etc/; tar -zxvf /u01/app/oracle/man_recovery_area/orcl/backup/tape/ob.tgz
sleep 3
service mhvtl start
service observiced start

lsscsi -g
obtool inventory -L lib01
lsscsi -g
```

7. Oracle11gR2 RAC 备份恢复案例一：完全恢复类场景批量模拟以及恢复要点

7.1 1a_users 表空间在线损坏

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 1 三个选项选 y。每个选项结束后，分别提供以下输出： “ftp://192.168.0.245/pub/bclresult-xx/prepare” “ftp://192.168.0.245/pub/bclresult-xx/real” “ftp://192.168.0.245/pub/bclresult-xx/end”下同
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 1a 1a_users 表空间在线损坏 sub_detecting sub_creating sub_offlining_users_immediate sub_destroying "users*" sub_revealing echo " " sub_clearing echo "THE END"

“sub_creating”调用的**“ctable.sql”**内容：	--drop table hr.edu234;
	--
	set echo off
	set feedback off
	select * from sys.tname;
	create table hr.edu234 (a number) tablespace users;
	alter system switch logfile;
	insert into hr.edu234 values(1);
	alter system switch logfile;
	commit;
	alter system switch logfile;
	insert into hr.edu234 values(2);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(3);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(4);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(5);
	commit;
	--
	set serveroutput on
	exec dbms_output.put_line(' ');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line('Table successfully created.');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line(' ');

恢复要点：	以下所有备份恢复实验，如果使用 rman，连接 catalog 与否，用户自选除非注明。 在 rman 中恢复：restore 该数据文件，recover 该数据文件。

7.2 1b_下线 user 表空间损坏

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 1b
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 1b 1b_下线 user 表空间损坏 sub_detecting sub_creating sub_offlining_users_nocatalog LABS-1B-USERS-AFTER-OFFLINE sub_destroying "users*" sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	在 rman 中恢复：restore 该数据文件，online 该表空间 。

7.3 1c_只读 user 表空间损坏

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 1c
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 1c 1c_只读 user 表空间损坏 sub_detecting sub_creating sub_readingonly_users_nocatalog LABS-1C-USERS-AFTER-READ-ONLY sub_offlining_users_immediate sub_destroying "users*" sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	在 rman 中恢复：restore 该数据文件，online 该表空间 ， read write 该表空间。

7.4 1d_users 表空间热备

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 1d
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 1d 1d_users 表空间热备 sub_detecting sub_creating echo " " echo " " sub_revealing sub_shutdowning_abort sub_clearing echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	--drop table hr.edu234; -- set echo off set feedback off select * from sys.tname; create table hr.edu234 (a number) tablespace users; alter system switch logfile;

	insert into hr.edu234 values(1);
	alter system switch logfile;
	commit;
	alter system switch logfile;
	insert into hr.edu234 values(2);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(3);
	alter system switch logfile;
	commit;
	set echo on
	set feedback on
	alter tablespace users begin backup;
	select * from v$backup;
	set echo off
	set feedback off
	insert into hr.edu234 values(4);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(5);
	commit;
	--
	set serveroutput on
	exec dbms_output.put_line(' ');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line('Table successfully created.');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line(' ');

恢复要点：	1）启动到 mount 在 sqlplus 中恢复： alter database end backup; alter database open; 2）srvctl 启动其他实例。

7.5 3_system 表空间离线损坏

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 3
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 3 3_system 表空间离线损坏 sub_detecting sub_creating sub_shutdowning_normal sub_destroying "system*" sub_revealing echo " " sub_clearing echo "THE END"

“sub_creating” 调用的 **“ctable.sql”**内容：	--drop table sys.edu234;
	--
	set echo off
	set feedback off
	select * from sys.tname;
	create table sys.edu234 (a number) tablespace system;
	alter system switch logfile;
	insert into sys.edu234 values(1);
	alter system switch logfile;
	commit;
	alter system switch logfile;
	insert into sys.edu234 values(2);
	alter system switch logfile;
	commit;
	insert into sys.edu234 values(3);
	alter system switch logfile;
	commit;
	insert into sys.edu234 values(4);
	alter system switch logfile;
	commit;
	insert into sys.edu234 values(5);

	commit;
	--
	set serveroutput on
	exec dbms_output.put_line(' ');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line('Table successfully created.');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line(' ');

恢复要点：	1）启动到 mount 在 rman 中恢复：restore system 表空间，recover system 表空间。 2）srvctl 启动其他实例。

7.6 4_tbsocp05_test 没有备份的表空间损坏

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 4
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 4 4_tbsocp05_test 没有备份的表空间损坏 sub_detecting echo "***TABLESPACE CREATED***" sub_creating sub_offlining_tbsocp05_test_immediate sub_destroying "tbsocp05_test*" sub_revealing echo " " sub_clearing echo "THE END"

“sub_creating” 调用的 **“ctable.sql”**内容：	--drop table hr.edu234;
	--
	set echo off
	set feedback off
	select * from sys.tname;
	create tablespace tbsocp05_test datafile size 5M;
	create table hr.edu234 (a number) tablespace tbsocp05_test;
	alter system switch logfile;
	insert into hr.edu234 values(1);
	alter system switch logfile;
	commit;
	alter system switch logfile;
	insert into hr.edu234 values(2);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(3);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(4);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(5);
	commit;
	--
	set serveroutput on
	exec dbms_output.put_line(' ');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line('Table successfully created.');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line(' ');

恢复要点：	1）在 sqlplus 里恢复：alter database create datafile '原文件名'; 2）OMF 改了文件名，因此：alter database rename file '旧文件名' to '新文件名'; 3）recover datafile '新文件名' ;

8. Oracle11gR2 RAC 备份恢复案例二：不完全恢复类场景批量模拟以及恢复要点

8.1 5_基于时间的不完全恢复

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 5
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 5 5_基于时间的不完全恢复 sub_detecting sub_creating echo " " echo " " sub_revealing echo " " sub_clearing echo "THE END"

“sub_creating”调用的**“ctable.sql”**内容：	--drop table hr.edu234;
	--
	set echo off
	set feedback off
	select * from sys.tname;
	create table hr.edu234 (a number) tablespace users;
	alter system switch logfile;
	insert into hr.edu234 values(1);
	alter system switch logfile;
	commit;
	--
	set serveroutput on
	exec dbms_output.put_line(' ');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line('Table successfully created.');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line(' ');
	--
	select * from hr.edu234;
	exec
	dbms_output.put_line('**
	********************************** ');
	exec dbms_output.put_line('After the displayed moment, the table will continuously be inserted with more rows,
	but finnally be droped.');
	exec
	dbms_output.put_line('**
	********************************** ');
	exec dbms_output.put_line(' ');
	host sleep 5
	alter session set nls_date_format='yyyy-mm-dd:hh24:mi:ss';
	select sysdate from dual;
	host sleep 3
	alter system switch logfile;
	insert into hr.edu234 values(2);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(3);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(4);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(5);
	commit;
	select * from hr.edu234;
	drop table hr.edu234 purge;

恢复要点：	1）启动到 mount 在 rman 的 run{}块中恢复：根据 ctable.sql 输出的时间 set until time；restore database；recover database；alter database open resetlogs; 2）srvctl 启动其他实例。

8.2 6a_基于 log 序列号的不完全恢复

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 6a
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 6a 6a_基于 log 序列号的不完全恢复 sub_detecting sub_creating echo " " echo " " sub_revealing echo " " sub_clearing echo "THE END"

“sub_creating”调用的**“ctable.sql”**内容：	--drop table hr.edu234;
	--
	set echo off
	set feedback off
	select * from sys.tname;
	create table hr.edu234 (a number) tablespace users;
	alter system switch logfile;
	insert into hr.edu234 values(1);
	alter system switch logfile;
	commit;
	--
	set serveroutput on
	exec dbms_output.put_line(' ');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line('Table successfully created.');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line(' ');
	--
	select * from hr.edu234;
	exec
	dbms_output.put_line('**********...********** ');
	exec dbms_output.put_line('After the displayed moment, the table will continuously be inserted with more rows, but finnally be droped..');
	exec
	dbms_output.put_line('**********...********** ');
	exec dbms_output.put_line(' ');
	host sleep 5
	select THREAD#, SEQUENCE# from v$log;
	archive log list
	host sleep 3
	alter system switch logfile;
	insert into hr.edu234 values(2);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(3);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(4);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(5);
	commit;
	select * from hr.edu234;
	drop table hr.edu234 purge;

恢复要点：	注意：如果刚做完实验 5，本场景之前，rever 环境 1）启动到 mount 在 rman 的 run{}块中恢复：根据 ctable.sql 输出的日志序列号 set until sequence xx thread x；restore database；recover database；alter database open resetlogs; 2）特别注意 ctable.sql 输出的日志序列号需要加 1，因为 until 是“不包含”的意思。 3）srvctl 启动其他实例。

8.3 6b_基于 cancel 的不完全恢复

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 6b
“bcl --RACGRID11 g13 6b”之后会自动运行的 **“real-post”**脚本内容（其他场景无本项）：	#!/bin/sh rm -f /root/tmp/bclcustom.sh for i in `ls /root/tmp/bclresult*` do v_botang_thread=$(grep -A 2 INSTANCE_NUMBER $i \| tail -n 1 \| tr -d '[:blank:]') v_botang_currlog=$(grep 'Current log sequence' $i \| tail -n 1 \| cut -c 29-) v_botang_currlogplus1=$[$v_botang_currlog + 1] v_botang_oldestlog=$(grep 'Oldest online log sequence' $i \| tail -n 1 \| cut -c 29-) v_botang_oldestlogbefore=$(grep 'Oldest online log sequence' $i \| head -n 1 \| cut -c 29-) v_botang_nextlog=$(grep 'Next log sequence to archive' $i \| tail -n 1 \| cut -c 29-) v_botang_nextlogbefore=$(grep 'Next log sequence to archive' $i \| head -n 1 \| cut -c 29-) v_botang_currlogdest=$(grep 'Archive destination' $i \| tail -n 1 \| cut -c 29-) v_botang_ip=$(echo $i | cut -f 2 -d -) perl -i -pe "s,Oldest online log sequence $v_botang_oldestlog,Oldest online log sequence XXX," $i perl -i -pe "s,Oldest online log sequence $v_botang_oldestlogbefore,Oldest online log sequence AAA," $i perl -i -pe "s,Next log sequence to archive$v_botang_nextlog,Next log sequence to archive YYY," $i perl -i -pe "s,Next log sequence to archive$v_botang_nextlogbefore,Next log sequence to archive BBB," $i grep -v 'Current log sequence' $i > $i.tmp rm -f $i mv $i.tmp $i cat > /root/tmp/rac.sh <<EOL if [-f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat] then su - grid -c "export ORACLE_SID=+ASM1;export ORACLE_HOME=/u01/app/11.2.0/grid;asmcmd"<<EOF > /home/oracle/asm find / thread_${v_botang_thread}_seq_$v_botang_currlogplus1* exit EOF sleep 1 v_path1=\$(cut -f 2 -d '+' /home/oracle/asm | head -n 1 | tr -d [:blank:]) v_path2="+"\$v_path1 su - grid -c "export ORACLE_SID=+ASM1;export ORACLE_HOME=/u01/app/11.2.0/grid;asmcmd"<<EOF rm -rf \$v_path2 exit EOF fi EOL scp -q -o ConnectTimeout=1 -o StrictHostKeyChecking=no -o NumberOfPasswordPrompts=1 -o ConnectionAttempts=1 /root/tmp/rac.sh $v_botang_ip:/usr/bin ssh -q -o ConnectTimeout=1 -o StrictHostKeyChecking=no -o NumberOfPasswordPrompts=1 -o ConnectionAttempts=1 $v_botang_ip "chmod +x /usr/bin/rac.sh" ssh -q -o ConnectTimeout=1 -o StrictHostKeyChecking=no -o NumberOfPasswordPrompts=1 -o ConnectionAttempts=1 $v_botang_ip /usr/bin/rac.sh >>$i 2>&1 ssh -q -o ConnectTimeout=1 -o StrictHostKeyChecking=no -o NumberOfPasswordPrompts=1 -o ConnectionAttempts=1 $v_botang_ip "rm -rf /usr/bin/rac.sh" echo "DELETE ARCHIVELOGS SUCCESSFULLY,THE END AGAIN" >> $i done
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 6b 6b_基于 cancel 的不完全恢复 sub_detecting sub_creating sub_shutdowning_normal sub_destroying "system*" sub_destroying "undotbs1*" sub_destroying "undotbs2*" sub_destroying "sysaux*" sub_destroying "example*" sub_destroying "users*" sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating” 调用的 **“ctable.sql”**内容：	同“6a_基于 log 序列号的不完全恢复”

恢复要点：	注意：如果刚做完实验 6a，本场景之前，rever 环境 1）归档日志被删除，日志序列号请用 asmcmd 确认。 2）为了练习 cancel 语法，本实验在 sqlplus 进行。 3）启动到 mount 在 rman 中做一些预备工作：restore 那些 delete input 的 archivelog；restore database; 4）在 sqlplus 中恢复：recover database until cancel; alter database open resetlogs; 5）srvctl 启动其他实例。

8.4 7a_当前控制文件和数据文件完好_日志文件全部损坏_正常关机不完全恢复_数据不丢_不需备份

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 7a
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 7a 7a_当前控制文件和数据文件完好_日志文件全部损坏_正常关机不完全恢复_数据不丢_不需备份 sub_detecting sub_creating sub_shutdowning_normal sub_destroying "group*" sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	1）启动到 mount 在 sqlplus 中恢复：recover database until cancel; alter database open resetlogs; 2）srvctl 启动其他实例。

8.5 7b_当前控制文件和数据文件完好_日志文件全部损坏_不正常关机不完全恢复

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 7b
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 7b 7b_当前控制文件和数据文件完好_日志文件全部损坏_不正常关机不完全恢复 sub_detecting sub_creating sub_shutdowning_abort sub_destroying "group*" sub_revealing echo " " sub_clearing echo 'From then on, catalog your database, but use it or not depends on you when recovering !!!' echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	注意：如果刚做完实验 7a，本场景之前，rever 环境 1）先按 7a 做，会失败。失败时会提示在线日志文件 current 组的 sequence 号。日志序列号请用 asmcmd 确认。 2）在 rman 的 run{}块中恢复：set until sequence 在线日志文件 current 组的 sequence 号 thread x ; restore database；recover database；alter database open resetlogs; 3）srvctl 启动其他实例。

8.6 8a_当前控制文件损坏_不完全恢复_用控制文件二进制备份_数据不丢_不需备份

在 **PXE** 推送端主机上运行：	注意：此编号之后的实验，请准备好 catalog。恢复目录存在于一台 IP 为 192.168.0.90 的主机上，服务名为“rcat.example.com”，监听端口为：“1521”。恢复目录用户名为“u90”，密码为“oracle_4U”。请事先准备好该设备。

	[root@server1 ~]# bcl --RACGRID11g13 8a
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 8a 8a_当前控制文件损坏_不完全恢复_用控制文件二进制备份_数据不丢_不需备份 sub_detecting sub_creating sub_backingup_controlfile_nocatalog LABS-8A-CONTROLFILE sub_resync_catalog sub_shutdowning_abort sub_destroying "current*" sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	1）启动到 nomount 在 rman 中恢复：restore controlfile；mount 数据库 2）在 rman 中恢复：recover database； alter database open resetlogs；（二进制控制文件 restore 后指导的恢复，都要 resetlogs） 4）srvctl 启动其他实例。

8.7 8b_当前控制文件损坏_完全恢复_用控制文件脚本_不需备份

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 8b
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 8b 8b_当前控制文件损坏_完全恢复_用控制文件脚本_不需备份 sub_detecting sub_creating sub_scripting_controlfile_nocatalog sub_deleting_backup_controlfile_nocatalog sub_shutdowning_abort sub_destroying "current*" sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	1）找到 sub_scripting_controlfile_nocatalog 给出的 trace 文件名，编辑成 sql 脚本：留下 set #1 noresetlogs 部分，去掉 set #2。 2）启动到 nomount 在 sqlplus 恢复：改 cluster_database 初始化参数为 false；执行以上脚本；改 cluster_database 初始化参数为 true；recover database；alter database open； 3）srvctl 启动其他实例。

8.8 9a_当前控制文件损坏_下线 user 表空间不完全恢复_用控制文件二进制备份_数据不丢_不需备份

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 9a
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 9a 9a_当前控制文件损坏_下线 user 表空间不完全恢复_用控制文件二进制备份_数据不丢_不需备份 sub_detecting sub_creating sub_backingup_controlfile_nocatalog LABS-9A-CONTROL-BEFORE-OFFLINE sub_offlining_users_nocatalog LABS-9A-USERS-AFTER-OFFLINE sub_backingup_controlfile_nocatalog LABS-9A-CONTROL-AFTER-OFFLINE

	sub_resync_catalog sub_shutdowning_abort sub_destroying "current*" sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	1）启动到 nomount 在 rman 中恢复：restore controlfile；mount 数据库；用 tag 为“LABS-9A-CONTROL-BEFORE-OFFLINE”或 tag 为“LABS-9A-CONTROL-AFTER-OFFLINE”的控制文件备份来恢复都可以。控制文件能经历表空间从上线到下线的变化。 2）在 rman 中恢复：recover database； alter database open resetlogs；（二进制控制文件 restore 后指导的恢复，都要 resetlogs）；alter tablespace users online； 4）srvctl 启动其他实例。

8.9 9b_当前控制文件损坏_下线 user 表空间完全恢复_用控制文件脚本_不需备份

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 9b
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 9b 9b_当前控制文件损坏_下线 user 表空间完全恢复_用控制文件脚本_不需备份 sub_detecting sub_creating echo "***BEFORE TABLESPACE OFFLINE***" sub_scripting_controlfile_nocatalog sub_offlining_users_nocatalog LABS-9B-USERS-AFTER-OFFLINE echo "***AFTER TABLESPACE OFFLINE***" sub_scripting_controlfile_nocatalog sub_deleting_backup_controlfile_nocatalog sub_resync_catalog sub_shutdowning_abort sub_destroying "current*" sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	1）找到 sub_scripting_controlfile_nocatalog 给出的 trace 文件名，编辑成 sql 脚本：留下 set #1 noresetlogs 部分，去掉 set #2。在 users 表空间下线前后各有一个 trace 文件，用之前的那个，不然会出现 missing00004。 2）启动到 nomount 在 sqlplus 恢复：改 cluster_database 初始化参数为 false；执行以上脚本；改 cluster_database 初始化参数为 true；recover database；alter database open；alter tablespace users online; 3）srvctl 启动其他实例。

8.10 9c_当前控制文件损坏_只读 user 表空间不完全恢复_用控制文件二进制备份_数据不丢_不需备份

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 9c
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 9c 9c_当前控制文件损坏_只读 user 表空间不完全恢复_用控制文件二进制备份_数据不丢_不需备份 sub_detecting sub_creating sub_backingup_controlfile_nocatalog LABS-9C-CONTR-BEFORE-READONLY sub_readingonly_users_nocatalog LABS-9C-USERS-AFTER-READONLY

	sub_backingup_controlfile_nocatalog LABS-9C-CONTROL-AFTER-READONLY sub_resync_catalog sub_shutdowning_abort sub_destroying "current*" sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	1）启动到 nomount 在 rman 中恢复：restore controlfile；mount 数据库；用 tag 为“LABS-9C-CONTR-BEFORE-READONLY”或 tag 为“LABS-9C-CONTROL-AFTER-READONLY”的控制文件备份来恢复都可以。控制文件能经历表空间从读写到只读的变化。 2）在 rman 中恢复：recover database； alter database open resetlogs；（二进制控制文件 restore 后指导的恢复，都要 resetlogs）；alter tablespace users online；alter tablespace users read write; 4）srvctl 启动其他实例。

8.11 9d_当前控制文件损坏_只读 user 表空间完全恢复_用控制文件脚本_不需备份

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 9d
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 9d 9d_当前控制文件损坏_只读 user 表空间完全恢复_用控制文件脚本_不需备份 sub_detecting sub_creating echo "***BEFORE TABLESPACE READONLY***" sub_scripting_controlfile_nocatalog sub_readingonly_users_nocatalog LABS-9D-USERS-AFTER-READ-ONLY echo "***AFTER TABLESPACE READONLY***" sub_scripting_controlfile_nocatalog sub_deleting_backup_controlfile_nocatalog sub_resync_catalog sub_shutdowning_abort sub_destroying "current*" sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	1）找到 sub_scripting_controlfile_nocatalog 给出的 trace 文件名，编辑成 sql 脚本：留下 set #1 noresetlogs 部分，去掉 set #2。在 users 表空间只读前后各有一个 trace 文件，用之前的那个，不然会出现 missing00004。 2）启动到 nomount 在 sqlplus 中恢复：改 cluster_database 初始化参数为 false；执行以上脚本；改 cluster_database 初始化参数为 true；recover database；alter database open；alter tablespace users online; alter tablespace users read write; 3）srvctl 启动其他实例。

8.12 10a_当前控制文件损坏_备份时下线 user 表空间不完全恢复_用控制文件二进制备份_数据不丢_不需备份

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 10a
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 10a 10a_当前控制文件损坏_备份时下线 user 表空间不完全恢复_用控制文件二进制备份_数据不丢_不需备份 sub_detecting sub_deleting_early_online_backup_users

	echo "***LABS TOTALLY DIFFERENT BEGINS: TABLESPACE OFFLINE***" echo "***BEFORE TABLESPACE ONLINE***" echo "***BACKUP***" sub_offlining_users_nocatalog LABS-10A-USERS-BEFORE-ONLINE sub_backingup_controlfile_nocatalog LABS-10A-CONTR-BEFORE-ONLINE echo "***FROM THEN ON: ONLINE***" sub_onlining_users_nocatalog LABS-10A-USERS-AFTER-ONLINE sub_backingup_controlfile_nocatalog LABS-10A-CONTR-AFTER-ONLINE sub_resync_catalog sub_creating sub_shutdowning_abort sub_destroying "current*" sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	1）启动到 nomount 在 rman 中恢复：restore controlfile；mount 数据库；用 tag 为“LABS-10A-CONTR-AFTER-ONLINE”的控制文件备份来恢复。这是因为控制文件不能很好地经历表空间从下线到上线的变化。如果用了 tag 为“LABS-10A-CONTR-BEFORE-ONLINE”的控制文件备份来恢复，resetlogs 之后需要跟上后续恢复。我们对比发现在不连接 catalog 的情况下进行恢复，10gR2 的数据库如果用错了以上的 tag，问题更严重，根本无法打开数据库。 2）在 rman 中恢复：recover database； alter database open resetlogs；（二进制控制文件 restore 后指导的恢复，都要 resetlogs）； 4）srvctl 启动其他实例。

8.13 10b_当前控制文件损坏_备份时下线 user 表空间完全恢复_用控制文件脚本_不需备份

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 10b
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 10b 10b_当前控制文件损坏_备份时下线 user 表空间完全恢复_用控制文件脚本_不需备份 sub_detecting sub_deleting_early_online_backup_users echo "***LABS TOTALLY DIFFERENT BEGINS: TABLESPACE OFFLINE***" echo "***BEFORE TABLESPACE ONLINE***" echo "***BACKUP***" sub_offlining_users_nocatalog LABS-10B-USERS-BEFORE-ONLINE sub_scripting_controlfile_nocatalog echo "***FROM THEN ON: ONLINE***" sub_onlining_users_nocatalog LABS-10B-USERS-AFTER-ONLINE sub_scripting_controlfile_nocatalog sub_deleting_backup_controlfile_nocatalog sub_resync_catalog sub_creating sub_shutdowning_abort sub_destroying "current*" sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	1）找到 sub_scripting_controlfile_nocatalog 给出的 trace 文件名，编辑成 sql 脚本：留下 set #1 noresetlogs 部分，去掉 set #2。在 users 表空间下线前后各有一个 trace 文件，用之后的那个，不然会出现 missing00004。 2）启动到 nomount 在 sqlplus 中恢复：改 cluster_database 初始化参数为 false；执行以上脚本；改 cluster_database 初始化参数为 true；recover database；alter database open； 3）srvctl 启动其他实例。

8.14 10c_当前控制文件损坏_备份时只读 user 表空间不完全恢复_用控制文件二进制备份_数据不丢_不需备份

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 10c
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 10c 10c_当前控制文件损坏_备份时只读 user 表空间不完全恢复_用控制文件二进制备份_数据不丢_不需备份 sub_detecting sub_deleting_early_online_backup_users echo "***LABS TOTALLY DIFFERENT BEGINS: TABLESPACE READ ONLY***" echo "***BEFORE TABLESPACE WRITE***" echo "***BACKUP***" sub_readingonly_users_nocatalog LABS-10C-USERS-BEFORE-RW sub_backingup_controlfile_nocatalog LABS-10C-CONTR-BEFORE-RW echo "***FROM THEN ON: READ WARITE***" sub_readingwrite_users_nocatalog LABS-10C-USERS-AFTER-RW sub_backingup_controlfile_nocatalog LABS-10C-CONTR-AFTER-RW sub_resync_catalog sub_creating sub_shutdowning_abort sub_destroying "current*" sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating” 调用的 **“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点**:**	1）启动到 nomount 在 rman 中恢复：restore controlfile；mount 数据库；用 tag 为“LABS-10C-CONTR-AFTER-RW”的控制文件备份来恢复。这是因为控制文件不能很好地经历表空间从下线到上线的变化。如果用了 tag 为“LABS-10C-CONTR-BEFORE-RW”来恢复，resetlogs 之后需要跟上后续恢复。我们对比发现在不连接 catalog 的情况下进行恢复，10gR2 的数据库如果用错了以上的 tag，问题更严重，根本无法打开数据库。 2）在 rman 中恢复：recover database； alter database open resetlogs；（二进制控制文件 restore 后指导的恢复，都要 resetlogs）； 4）srvctl 启动其他实例。

8.15 10d_当前控制文件损坏_备份时只读 user 表空间完全恢复_用控制文件脚本_不需备份

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 10d
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 10d 10d_当前控制文件损坏_备份时只读 user 表空间完全恢复_用控制文件脚本_不需备份 sub_detecting sub_deleting_early_online_backup_users echo "***LABS TOTALLY DIFFERENT BEGINS: TABLESPACE READ ONLY***" echo "***BEFORE TABLESPACE READ WRITE***" echo "***BACKUP***" sub_readingonly_users_nocatalog LABS-10D-USERS-BEFORE-RW sub_scripting_controlfile_nocatalog echo "***FROM THEN ON: READ WRITE***" sub_readingwrite_users_nocatalog LABS-10D-USERS-AFTER-RW sub_scripting_controlfile_nocatalog sub_deleting_backup_controlfile_nocatalog sub_resync_catalog sub_creating sub_shutdowning_abort sub_destroying "current*" sub_revealing echo " " sub_clearing

	echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	1）找到 sub_scripting_controlfile_nocatalog 给出的 trace 文件名，编辑成 sql 脚本：留下 set #1 noresetlogs 部分，去掉 set #2。在 users 表空间下线前后各有一个 trace 文件，用之后的那个，不然会出现 missing00004。 2）启动到 nomount 在 sqlplus 中恢复：改 cluster_database 初始化参数为 false；执行以上脚本；改 cluster_database 初始化参数为 true；recover database；alter database open； 3）srvctl 启动其他实例。

9. Oracle11gR2 RAC 备份恢复案例三：进阶不完全恢复类场景批量模拟以及恢复要点

9.1 11a_当前控制文件和日志文件全部损坏_不完全恢复_用控制文件二进制备份

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 11a
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 11a 11a_当前控制文件和日志文件全部损坏_不完全恢复_用控制文件二进制备份 sub_detecting sub_creating sub_backingup_controlfile_nocatalog LABS-11A-CONTROLFILE sub_resync_catalog sub_shutdowning_abort sub_destroying "current*" sub_destroying "group*" sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	注意：如果刚做完实验 10d，本场景之前，rever 环境 1）在线日志被删除，日志序列号请用 asmcmd 确认。 2）启动到 nomount 在 rman 中恢复：restore controlfile；mount 数据库。 3）在 rman 的 run{}中恢复：set until sequence 在线日志 current 组 sequence 号 thread x；restore database；recover database；alter database open resetlogs; 4) srvctl 启动其他实例。

9.2 11b_当前控制文件和日志文件全部损坏_不完全恢复_用控制文件脚本

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 11b
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 11b 11b_当前控制文件和日志文件全部损坏_不完全恢复_用控制文件脚本 sub_detecting sub_creating sub_scripting_controlfile_nocatalog sub_deleting_backup_controlfile_nocatalog sub_resync_catalog sub_shutdowning_abort sub_destroying "current*" sub_destroying "group*" sub_revealing echo " " sub_clearing echo "THE END"

“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	注意：如果刚做完实验 11a，本场景之前，rever 环境 1）找到 sub_scripting_controlfile_nocatalog 给出的 trace 文件名，编辑成 sql 脚本：留下 set #2 resetlogs 部分，去掉 set #1。 2）启动到 nomount 在 sqlplus 恢复：改 cluster_database 初始化参数为 false；执行以上脚本；改 cluster_database 初始化参数为 true；自动到 mount。 3）在 rman 中做一些预备工作：restore 那些 delete input 的 archivelog；restore database; 4）在线日志被删除，日志序列号请用 asmcmd 确认。 5）在 sqlplus 中恢复：recover database until cancel using backup controlfile; 注意 set #2 脚本中本身给的提示“recover database using backup controlfile”是不正确的，会直接导致打不开数据库。 6）在 sqlplus 中恢复： 遇到在线日志 current 组 sequence 号时，敲 cancel；alter database open resetlogs; 7）添加临时文件，块跟踪文件等。 8）srvctl 启动其他实例。

9.3 12a_当前控制文件和日志文件全部损坏_备份时下线 user 表空间不完全恢复_用控制文件二进制备份

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 12a
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 12a 12a_当前控制文件和日志文件全部损坏_备份时下线 user 表空间不完全恢复_用控制文件二进制备份 sub_detecting sub_deleting_early_online_backup_users echo "***LABS TOTALLY DIFFERENT BEGINS: TABLESPACE OFFLINE***" echo "***BEFORE TABLESPACE ONLINE***" echo "***BACKUP***" sub_offlining_users_nocatalog LABS-12A-USERS-BEFORE-ONLINE sub_backingup_controlfile_nocatalog LABS-12A-CONTR-BEFORE-ONLINE echo "***FROM THEN ON: ONLINE***" sub_onlining_users_nocatalog LABS-12A-USERS-AFTER-ONLINE sub_backingup_controlfile_nocatalog LABS-12A-CONTR-AFTER-ONLINE sub_resync_catalog sub_creating sub_shutdowning_abort sub_destroying "current*" sub_destroying "group*" sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	注意：如果刚做完实验 11b，本场景之前，rever 环境 1）先按 10a 做，会失败。在线日志被删除，日志序列号请用 asmcmd 确认。 2）与 10a 不同的是必需在 rman 的 run{}块中恢复：set until sequence 在线日志 current 组 sequence 号 thread x；restore database；recover database；alter database open resetlogs; 3）srvctl 启动其他实例。

9.4 12b_当前控制文件和日志文件全部损坏_备份时下线 user 表空间不完全恢复_用控制文件脚本

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 12b
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 12b 12b_当前控制文件和日志文件全部损坏_备份时下线 user 表空间不完全恢复_用控制文件脚本

	sub_detecting sub_deleting_early_online_backup_users echo "***LABS TOTALLY DIFFERENT BEGINS: TABLESPACE OFFLINE***" echo "***BEFORE TABLESPACE ONLINE***" echo "***BACKUP***" sub_offlining_users_nocatalog LABS-12B-USERS-BEFORE-ONLINE sub_scripting_controlfile_nocatalog echo "***FROM THEN ON: ONLINE***" sub_onlining_users_nocatalog LABS-12B-USERS-AFTER-ONLINE sub_scripting_controlfile_nocatalog sub_deleting_backup_controlfile_nocatalog sub_resync_catalog sub_creating sub_shutdowning_abort sub_destroying "current*" sub_destroying "group*" sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating” 调用的 **“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	注意：如果刚做完实验 12a，本场景之前，rever 环境 1）先按 10b 做，会失败。在线日志被删除，日志序列号请用 asmcmd 确认。 2）找到 sub_scripting_controlfile_nocatalog 给出的 trace 文件名，编辑成 sql 脚本：留下 set #2 resetlogs 部分，去掉 set #1。在 users 表空间下线前后各有一个 trace 文件，用之后的那个，不然会出现 missing00004。 3）启动到 nomount 在 sqlplus 恢复：改 cluster_database 初始化参数为 false；执行以上脚本；改 cluster_database 初始化参数为 true；自动到 mount。 4）在 rman 中做一些预备工作：restore 那些 delete input 的 archivelog；restore database; 5）在 sqlplus 中恢复：recover database until cancel using backup controlfile; 注意 set #2 脚本中本身给的提示“recover database using backup controlfile”是不正确的，会直接导致打不开数据库。 6）在 sqlplus 中恢复： 遇到在线日志 current 组 sequence 号时，敲 cancel；alter database open resetlogs; 7）添加临时文件，块跟踪文件等。 8）srvctl 启动其他实例。

9.5 12c_当前控制文件和日志文件全部损坏_备份时只读 user 表空间不完全恢复_用控制文件二进制备份

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 12c
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 12c 12c_当前控制文件和日志文件全部损坏_备份时只读 user 表空间不完全恢复_用控制文件二进制备份 sub_detecting sub_deleting_early_online_backup_users echo "***LABS TOTALLY DIFFERENT BEGINS: TABLESPACE READ ONLY***" echo "***BEFORE TABLESPACE WRITE***" echo "***BACKUP***" sub_readingonly_users_nocatalog LABS-12C-USERS-BEFORE-RW sub_backingup_controlfile_nocatalog LABS-12C-CONTR-BEFORE-RW echo "***FROM THEN ON: READ WARITE***" sub_readingwrite_users_nocatalog LABS-12C-USERS-AFTER-RW sub_backingup_controlfile_nocatalog LABS-12C-CONTR-AFTER-RW sub_resync_catalog sub_creating sub_shutdowning_abort sub_destroying "current*" sub_destroying "group*" sub_revealing echo " " sub_clearing echo "THE END"

“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	注意：如果刚做完实验 12b，本场景之前，rever 环境 1）先按 10c 做，会失败。在线日志被删除，日志序列号请用 asmcmd 确认。 2）与 10c 不同的是必需在 rman 的 run{}块中恢复：set until sequence 在线日志 current 组 sequence 号 thread x；restore database；recover database; alter database open resetlogs; 3）srvctl 启动其他实例。

9.6 12d_当前控制文件和日志文件全部损坏_备份时只读 user 表空间不完全恢复_用控制文件脚本

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 12d
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 12d 12d_当前控制文件和日志文件全部损坏_备份时只读 user 表空间不完全恢复_用控制文件脚本 sub_detecting sub_deleting_early_online_backup_users echo "***LABS TOTALLY DIFFERENT BEGINS: TABLESPACE READ ONLY***" echo "***BEFORE TABLESPACE READ WRITE***" echo "***BACKUP***" sub_readingonly_users_nocatalog LABS-12D-USERS-BEFORE-RW sub_scripting_controlfile_nocatalog echo "***FROM THEN ON: READ WRITE***" sub_readingwrite_users_nocatalog LABS-12D-USERS-AFTER-RW sub_scripting_controlfile_nocatalog sub_deleting_backup_controlfile_nocatalog sub_resync_catalog sub_creating sub_shutdowning_abort sub_destroying "current*" sub_destroying "group*" sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	注意：如果刚做完实验 12c，本场景之前，rever 环境 1）先按 10d 做，会失败。在线日志被删除，日志序列号请用 asmcmd 确认。 2）找到 sub_scripting_controlfile_nocatalog 给出的 trace 文件名，编辑成 sql 脚本：留下 set #2 resetlogs 部分，去掉 set #1。在 users 表空间下线前后各有一个 trace 文件，用之后的那个，不然会出现 missing00004。 3）启动到 nomount 在 sqlplus 恢复：改 cluster_database 初始化参数为 false；执行以上脚本；改 cluster_database 初始化参数为 true；自动到 mount。 4）在 rman 中做一些预备工作：restore 那些 delete input 的 archivelog；restore database; 5）在 sqlplus 中恢复：recover database until cancel using backup controlfile; 注意 set #2 脚本中本身给的提示“recover database using backup controlfile”是不正确的，会直接导致打不开数据库。 6）在 sqlplus 中恢复： 遇到在线日志 current 组 sequence 号时，敲 cancel；alter database open resetlogs; 7）添加临时文件，块跟踪文件等。 8）srvctl 启动其他实例。

9.7 13a_当前控制文件损坏_新建 tbsocp05_test2 表空间不完全恢复_用控制文件二进制备份_数据不丢_不需备份

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 13a
场景模拟脚本	. /home/oracle/bclcustom-subprogram

内容：	sub_getting 13a 13a_当前控制文件损坏_新建 tbsocp05_test2 表空间不完全恢复_用控制文件二进制备份_数据不丢_不需备份 sub_detecting echo "BEFORE TABLESPACE CREATION" echo "***BACKUP***" sub_backingup_controlfile_nocatalog LABS-13A-CONTR-BEFORE-CREA sub_resync_catalog echo "***TABLESPACE CREATED***" sub_creating echo "AFTER TABLESPACE CREATION" echo "***BACKUP AGAIN***" sub_backingup_controlfile_nocatalog LABS-13A-CONTR-AFTER-CREA sub_resync_catalog sub_shutdowning_abort sub_destroying "current*" sub_revealing echo " " sub_clearing echo "THE END"

“sub_creating” 调用的 **“ctable.sql”**内容：	--drop table hr.edu234;
	--
	set echo off
	set feedback off
	select * from sys.tname;
	create tablespace tbsocp05_test2 datafile size 5M;
	create table hr.edu234 (a number) tablespace tbsocp05_test2;
	alter system switch logfile;
	insert into hr.edu234 values(1);
	alter system switch logfile;
	commit;
	alter system switch logfile;
	insert into hr.edu234 values(2);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(3);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(4);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(5);
	commit;
	--
	set serveroutput on
	exec dbms_output.put_line(' ');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line('Table successfully created.');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line(' ');

恢复要点：	1）启动到 nomount 在 rman 中恢复：restore controlfile；mount 数据库；用 tag 为“LABS-13A-CONTR-BEFORE-CREA”或 tag 为“LABS-13A-CONTR-AFTER-CREA”的控制文件备份来恢复都可以。控制文件能经历新建表空间的变化。我们对比发现在不连接 catalog 的情况下进行恢复，10gR2 的数据库如果用 tag 为“LABS-13A-CONTR-BEFORE-CREA”来恢复，该新建表空间对应的数据文件会出现 uname00007 问题，需要跟上后续恢复。 2）在 rman 中恢复：recover database； alter database open resetlogs；（二进制控制文件 restore 后指导的恢复，都要 resetlogs） 4）srvctl 启动其他实例。

9.8 13b_当前控制文件损坏_新建 tbsocp05_test3 表空间完全恢复_用控制文件脚本_不需备份

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 13b
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 13b 13b_当前控制文件损坏_新建 tbsocp05_test3 表空间完全恢复_用控制文件脚本_不需备份 sub_detecting echo "BEFORE TABLESPACE CREATION"

	sub_scripting_controlfile_nocatalog echo "***TABLESPACE CREATED***" sub_creating echo "AFTER TABLESPACE CREATION" sub_scripting_controlfile_nocatalog sub_deleting_backup_controlfile_nocatalog sub_shutdowning_abort sub_destroying "current*" sub_revealing echo " " sub_clearing echo "THE END"

“sub_creating”调用的**“ctable.sql”**内容：	--drop table hr.edu234;
	--
	set echo off
	set feedback off
	select * from sys.tname;
	create tablespace tbsocp05_test3 datafile size 5M;
	create table hr.edu234 (a number) tablespace tbsocp05_test3;
	alter system switch logfile;
	insert into hr.edu234 values(1);
	alter system switch logfile;
	commit;
	alter system switch logfile;
	insert into hr.edu234 values(2);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(3);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(4);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(5);
	commit;
	--
	set serveroutput on
	exec dbms_output.put_line(' ');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line('Table successfully created.');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line(' ');

恢复要点**:**	1）找到 sub_scripting_controlfile_nocatalog 给出的 trace 文件名，编辑成 sql 脚本：留下 set #1 noresetlogs 部分，去掉 set #2。在新建表空间前后各有一个 trace 文件，用之后的那个，不然会出现 missing00004。 2）启动到 nomount 在 sqlplus 中恢复：改 cluster_database 初始化参数为 false；执行以上脚本；改 cluster_database 初始化参数为 true；recover database；alter database open； 3）srvctl 启动其他实例。

9.9 14_删除表空间不完全恢复

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 14
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 14 14_删除表空间不完全恢复 sub_detecting sub_altering_default_tablespace echo "***BEFORE TABLESPACE DROP***" echo "***BACKUP***" sub_backingup_controlfile_nocatalog LABS-14-CONTR-BEFORE-DROP sub_creating sub_dropping_users echo "***AFTER TABLESPACE DROP***" echo "***BACKUP AGAIN***" sub_backingup_controlfile_nocatalog LABS-14-CONTR-AFTER-DROP sub_resync_catalog echo " "

	sub_revealing echo " " sub_clearing echo "THE END"
“sub_creating”调用的**“ctable.sql”**内容：	同“1a_users 表空间在线损坏”
恢复要点：	注意：如果刚做完实验 13b，本场景之前，rever 环境 1）启动到 nomount 在 rman 中恢复：restore controlfile；mount 数据库；用 tag 为“LABS-14-CONTR-BEFORE-DROP”的控制文件备份来恢复。不然恢复后数据文件出 missing00004。 2）找所有实例上的 alert 日志，发现开始删除表空间的时间。将这个时间减去 1 秒。 3）启动到 mount 在 rman 的 run{}块中恢复：根据减去 1 秒的时间 set until time；restore database；recover database；alter database open resetlogs; 2）srvctl 启动其他实例。

9.10 15_穿越 incarnation 不完全恢复

在 **PXE** 推送端主机上运行：	[root@server1 ~]# bcl --RACGRID11g13 15
“bcl --RACGRID11 g13 15”之后会自动运行的**“real-post”**脚本内容（其他场景无本项）：	#!/bin/sh rm -f /root/tmp/bclcustom.sh for i in `ls /root/tmp/bclresult*` do v_botang_thread=$(grep -A 2 INSTANCE_NUMBER $i \| tail -n 1 \| tr -d '[:blank:]') v_botang_currlog=$(grep 'Current log sequence' $i \| tail -n 1 \| cut -c 29-) v_botang_currlogplus1=$[$v_botang_currlog + 1] v_botang_oldestlog=$(grep 'Oldest online log sequence' $i \| tail -n 1 \| cut -c 29-) v_botang_oldestlogbefore=$(grep 'Oldest online log sequence' $i \| head -n 1 \| cut -c 29-) v_botang_nextlog=$(grep 'Next log sequence to archive' $i \| tail -n 1 \| cut -c 29-) v_botang_nextlogbefore=$(grep 'Next log sequence to archive' $i \| head -n 1 \| cut -c 29-) v_botang_currlogdest=$(grep 'Archive destination' $i \| tail -n 1 \| cut -c 29-) v_botang_ip=$(echo $i | cut -f 2 -d -) perl -i -pe "s,Oldest online log sequence $v_botang_oldestlog,Oldest online log sequence XXX," $i perl -i -pe "s,Oldest online log sequence $v_botang_oldestlogbefore,Oldest online log sequence AAA," $i perl -i -pe "s,Next log sequence to archive$v_botang_nextlog,Next log sequence to archive YYY," $i perl -i -pe "s,Next log sequence to archive$v_botang_nextlogbefore,Next log sequence to archive BBB," $i grep -v 'Current log sequence' $i > $i.tmp rm -f $i mv $i.tmp $i cat > /root/tmp/rac.sh <<EOL v_dbname=\$(cat /home/oracle/dbname | tr -d [:blank:]) v_ip=\$(ifconfig | grep '192\.168\.0' | head -n 1| cut -d . -f 4| cut -d ' ' -f 1) grep -q 'LABS ADD' /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora if [\$? == '1'] then echo " " >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora echo "rcat =" >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora echo " (DESCRIPTION =" >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora echo " (ADDRESS = (PROTOCOL = TCP)(HOST = 192.168.0.90)(PORT = 1521))" >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora echo " (CONNECT_DATA =" >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora echo " (SERVER = DEDICATED)" >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora echo " (SERVICE_NAME = rcat.example.com)" >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora echo ")" >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora echo ")" >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora echo " " >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora echo '#LABS ADD' >> /u01/app/oracle/product/11.2.0/dbhome_1/network/admin/tnsnames.ora fi if [-f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat] then echo "***Connecting Catalog***" su - oracle -c "rman target sys/oracle_4U@\$v_dbname catalog u90/oracle_4U@rcat" <<! resync catalog;

```
!
 fi
 echo " "
 if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]
 then
  su - grid -c "export ORACLE_SID=+ASM1;export ORACLE_HOME=/u01/app/11.2.0/grid;asmcmd"<<EOF
> /home/oracle/asm
  find / thread_${v_botang_thread}_seq_$v_botang_currlogplus1*
  exit
EOF
  sleep 1
  v_path1=\$(cut -f 2 -d '+' /home/oracle/asm  | head -n 1 | tr -d [:blank:])
  v_path2="+"\$v_path1
  su - grid -c "export ORACLE_SID=+ASM1;export ORACLE_HOME=/u01/app/11.2.0/grid;asmcmd"<<EOF
  rm -rf \$v_path2
  exit
EOF
 fi
EOL
 scp  -q -o ConnectTimeout=1  -o StrictHostKeyChecking=no -o NumberOfPasswordPrompts=1 -o
ConnectionAttempts=1 /root/tmp/rac.sh   $v_botang_ip:/usr/bin
 ssh -q -o ConnectTimeout=1  -o StrictHostKeyChecking=no -o NumberOfPasswordPrompts=1 -o
ConnectionAttempts=1 $v_botang_ip "chmod +x /usr/bin/rac.sh"
 ssh -q -o ConnectTimeout=1  -o StrictHostKeyChecking=no -o NumberOfPasswordPrompts=1 -o
ConnectionAttempts=1 $v_botang_ip /usr/bin/rac.sh >>$i 2>&1
 ssh -q -o ConnectTimeout=1  -o StrictHostKeyChecking=no -o NumberOfPasswordPrompts=1 -o
ConnectionAttempts=1 $v_botang_ip "rm -rf /usr/bin/rac.sh"
 rm -rf   /root/tmp/bclshell-$v_botang_ip
 echo "v_dbname=\$(cat /home/oracle/dbname | tr -d [:blank:])" >> /root/tmp/bclshell-$v_botang_ip
 echo 'su - oracle -c "srvctl start instance -i ${v_dbname}1 -d $v_dbname -o mount &>/dev/null"' >>
/root/tmp/bclshell-$v_botang_ip
 echo 'su - oracle -c "srvctl start instance -i ${v_dbname}2 -d $v_dbname -o mount &>/dev/null"' >>
/root/tmp/bclshell-$v_botang_ip
 echo "if [ -f /u01/app/11.2.0/grid/dbs/hc_+ASM1.dat ]" >> /root/tmp/bclshell-$v_botang_ip
 echo "then" >> /root/tmp/bclshell-$v_botang_ip
 echo "su - oracle -c \"rman target sys/oracle_4U@\${v_dbname} \" <<EOF " >> /root/tmp/bclshell-
$v_botang_ip
 echo "run { "   >> /root/tmp/bclshell-$v_botang_ip
 echo "set until sequence $v_botang_currlogplus1 thread $v_botang_thread;" >>  /root/tmp/bclshell-$v_botang_ip
 echo "restore database;"  >>  /root/tmp/bclshell-$v_botang_ip
 echo "recover database;"  >>  /root/tmp/bclshell-$v_botang_ip
 echo "}" >>  /root/tmp/bclshell-$v_botang_ip
 echo "EOF" >> /root/tmp/bclshell-$v_botang_ip
 echo "su - oracle -c \"sqlplus /nolog \" <<EOF " >> /root/tmp/bclshell-$v_botang_ip
 echo "conn  sys/oracle_4U@\${v_dbname}1 as sysdba"  >> /root/tmp/bclshell-$v_botang_ip
 echo "alter database flashback off;"  >>  /root/tmp/bclshell-$v_botang_ip
 echo "alter database open resetlogs;"  >>  /root/tmp/bclshell-$v_botang_ip
 echo "set echo off" >> /root/tmp/bclshell-$v_botang_ip
 echo "set feedback off" >> /root/tmp/bclshell-$v_botang_ip
 echo "@/home/oracle/rtable.sql;" >> /root/tmp/bclshell-$v_botang_ip
 echo "quit"  >> /root/tmp/bclshell-$v_botang_ip
 echo "EOF" >> /root/tmp/bclshell-$v_botang_ip
 echo "su - oracle -c \"sqlplus /nolog \" <<EOF " >> /root/tmp/bclshell-$v_botang_ip
 echo "conn  sys/oracle_4U@\${v_dbname}2 as sysdba"  >> /root/tmp/bclshell-$v_botang_ip
 echo "alter database open;"  >>  /root/tmp/bclshell-$v_botang_ip
 echo "quit"  >> /root/tmp/bclshell-$v_botang_ip
 echo "EOF" >> /root/tmp/bclshell-$v_botang_ip
 echo  "fi" >> /root/tmp/bclshell-$v_botang_ip
 echo "rm -f /home/oracle/rtable.sql" >> /root/tmp/bclshell-$v_botang_ip
 echo " echo \"DELETE ARCHIVELOGS HAVE ALREADY BEEN RECOVERED FOR LAB15
SUCCESSFULLY,THE END AGAIN\"" >> /root/tmp/bclshell-$v_botang_ip
 scp -q -o ConnectTimeout=1  -o StrictHostKeyChecking=no -o NumberOfPasswordPrompts=1 -o
ConnectionAttempts=1 /root/tmp/bclshell-$v_botang_ip    $v_botang_ip:/usr/bin/ >/dev/null 2>&1
 scp -q -o ConnectTimeout=1  -o StrictHostKeyChecking=no -o NumberOfPasswordPrompts=1 -o
ConnectionAttempts=1 /root/tmp/bclsend-$v_botang_ip/rtable.sql    $v_botang_ip:/home/oracle/ >/dev/null 2>&1
 ssh -q -o ConnectTimeout=1  -o StrictHostKeyChecking=no -o NumberOfPasswordPrompts=1 -o
ConnectionAttempts=1 $v_botang_ip  chmod +x /usr/bin/bclshell-$v_botang_ip
 (ssh -q -o ConnectTimeout=1  -o StrictHostKeyChecking=no -o NumberOfPasswordPrompts=1 -o
ConnectionAttempts=1 $v_botang_ip /usr/bin/bclshell-$v_botang_ip  >>$i 2>&1)&
```

	done
场景模拟脚本内容：	. /home/oracle/bclcustom-subprogram sub_getting 15 15_穿越 incarnation 不完全恢复 sub_detecting sub_creating sub_shutdowning_normal sub_destroying "system*" sub_destroying "undotbs1*" sub_destroying "undotbs2*" sub_destroying "sysaux*" sub_destroying "example*" sub_destroying "users*" sub_revealing echo " " sub_startuping_node1mount sub_clearing_excludertable echo "THE END"

“sub_creating”调用的**“ctable.sql”**内容：	--drop table hr.edu234;
	--
	set echo off
	set feedback off
	select * from sys.tname;
	select instance_number from v$instance;
	create table hr.edu234 (a number) tablespace users;
	create table hr.specialedu234(a varchar2(100)) tablespace users;
	insert into hr.specialedu234 values('11g SUCCESS');
	commit;
	host sleep 5
	alter session set nls_date_format='yyyy-mm-dd:hh24:mi:ss';
	select sysdate from dual;
	host sleep 3
	truncate table hr.specialedu234;
	alter system switch logfile;
	insert into hr.edu234 values(1);
	alter system switch logfile;
	commit;
	--
	set serveroutput on
	exec dbms_output.put_line(' ');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line('Table successfully created.');
	exec dbms_output.put_line('************************** ');
	exec dbms_output.put_line(' ');
	--
	select * from hr.edu234;
	exec
	dbms_output.put_line('**
	********************************** ');
	exec dbms_output.put_line('After the displayed moment, the table will continuously be inserted with more rows,
	but finnally be droped..');
	exec
	dbms_output.put_line('**
	********************************** ');
	exec dbms_output.put_line(' ');
	host sleep 5
	archive log list
	host sleep 3
	alter system switch logfile;
	insert into hr.edu234 values(2);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(3);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(4);
	alter system switch logfile;
	commit;
	insert into hr.edu234 values(5);

	commit; select * from hr.edu234; drop table hr.edu234 purge;
恢复要点：	注意：如果刚做完实验 14，本场景之前，rever 环境 1）启动到 mount 在 rman 中：reset database to incarnation 2; 2）在 rman 的 run{}块中恢复：根据 ctable.sql 输出的时间 set until time；restore database；recover database；alter database open resetlogs; 2）srvctl 启动其他实例。

总结

“推送+场景模拟”是本文的设计思想，希望对 DBA 演练备份恢复技术会有帮助。本文中所提及的所有原创程序均提供下载和 md5sum 文件。 读者使用这些程序不应出于商业目的，作者对使用这些程序可能带来的一切后果不承担任何法律责任。“bcp”和“bcl”的其他选项也能推送 10g 单机版环境、11g 单机版环境和 10gRAC 环境，同时也能模拟 10g 单机版环境、11g 单机版环境和 10gRAC 环境的备份恢复场景。请读者自行摸索这些功能。