Университет ИТМО

Кафедра ВТ

Системы управления базами данных.

Лабораторная работа #4.

Вариант 330.

Выполнили: Ванин Алексей Борисович, гр. P3401

Дидин Евгений Витальевич, гр. Р3401

Преподаватели:

Цопа Е.А.

Харитонова А.Е.

Санкт-Петербург, 2015 г.

Исходный текст задания:

Этап 1. Сконфигурировать кластер на платформе Oracle RAC, состоящий из двух узлов, размещённых на отдельных виртуальных машинах и создать кластерную базу данных с заданными параметрами.

Особенности конфигурации кластера и БД:

* Имя базы данных: "apprehensivedragoon".
* Имена узлов кластера: "apprehensivedragoon0" и "apprehensivedragoon1".
* В качестве хранилища необходимо использовать ASM, сконфигурированный в результате выполнения лабораторной работы №3. Хранилище ASM необходимо расположить на диске, доступном со всех узлов кластера по протоколу NFS.

Этап 2. Создать тестовые таблицы с записями и произвести следующие операции с БД:

* Создать нового пользователя "delightedbroodling".
* Закрыть все сессии на всех узлах кластера, запущенные от имени пользователя "delightedbroodling".
* Создать нового пользователя "exciteddrone".
* Закрыть все сессии на всех узлах кластера, запущенные от имени пользователя "delightedbroodling".
* Создать новую сессию от имени пользователя "exciteddrone".

Этап 3. Осуществить "внештатную" остановку узла кластера "apprehensivedragoon1", проверив таким образом, что вся нагрузка будет перенесена на узел "apprehensivedragoon0" и целостность данных не будет нарушена.

Этап 4. Выполнить ряд операций в следующей последовательности:

* Добавить Voting Disk (в виде файла) по пути $ASM\_HOME/intriguedwraith/.
* Удалить с помощью инструментов Oracle RAC созданный на предыдущем шаге Voting Disk.

Выполнение работы:

Для выполнения данной работы на основе виртуальных машин в среде Oracle VirtualBox был создан кластер из двух компьютеров под управлением ОС Oracle Linux 6.7 Они были сконфигурированы таким образом, чтобы имели доступ друг к другу через два сетевых устройства (внутренняя сеть и виртуальная сеть).

Установка происходит параллельно: устанавливая grid и database на одну ноду, автоматически происходит копирование файлов по сети на вторую ноду. Во время установки можно сконфигурировать имена нод согласно заданию.

После установки всех компонентов попробуем создать базу данных. Для работы в RAC наша база данных будет работать на двух инстансах (по одной на каждой ноде), и имена этих инстансов должны отличатся (цифровой индекс). Поэтому у каждой ноды будет свой уникальный init.ora

db\_block\_size=8192

memory\_target=300M

cluster\_database\_instances=2

db\_name=apprehensivedragoon

db\_create\_file\_dest=+DATA

apprehensivedragoon0.instance\_number=1

apprehensivedragoon1.instance\_number=2

Также по заданию следует настроить общую дисковую группу ASM, которую разделяют оба инстанса. В нашем случае эту работу возьмёт на себя отдельный внешний диск, подключённый к виртуальным машинам в режиме shared. Настроим его для работы дисковой группы ASM:

# oracleasm createdisk DISK1 /dev/sdb1

Writing disk header: done

Instantiating disk: done

# oracleasm scandisks

Reloading disk partitions: done

Cleaning any stale ASM disks...

Scanning system for ASM disks...

# oracleasm listdisks

DISK1

Команда oracleasm listdisks выполняется одинаково с обеих нод, что говорит о том, что разделяемый диск настроен корректно.

Также для работы следует установить слушателей для работы с подключениями. Отредактируем нужным образом файл tnsnames.ora

APPREHENSIVEDRAGOON =

 (DESCRIPTION =

 (ADDRESS = (PROTOCOL = TCP)(HOST = apprehensivedragoon-scan.localdomain)(PORT = 1521))

 (CONNECT\_DATA =

 (SERVER = DEDICATED)

 (SERVICE\_NAME = apprehensivedragoon.localdomain)

 )

 )

Во время установки для корректного сетевого взаимодействия также был скорректирован файл /etc/hosts для правильной отправки сообщений по внутренним сетям между нодами.

После начальной настройки создаём базу данных таким же образом как в первых лабораторных работах.

create database apprehensivedragoon

logfile group 1 ('+DATA/redo1.log') size 100M,

group 2 ('+DATA/redo2.log') size 100M

character set WE8ISO8859P1

national character set utf8

datafile

'+DATA/datafile/system' size 200M

sysaux

datafile

'+DATA//datafile/sysaux' size 100M reuse

default tablespace users

datafile

'+DATA/datafile/users' size 100M reuse

undo tablespace undotbs1

datafile

'+DATA/datafile/undotbs1' size 100M,

'+DATA/datafile/undotbs2' size 100M;

После создания БД можно включить её в кластерный режим работы.

$ srvctl add database -d apprehensivedragoon -o $ORACLE\_HOME

$ srvctl add instance -d apprehensivedragoon -i apprehensivedragoon1 -n apprehensivedragoon0

$ srvctl add instance -d apprehensivedragoon -i apprehensivedragoon2 -n apprehensivedragoon1

Далее можно протестировать кластерную работу. Здесь и далее команды начинающиеся с SQL 1> исполняются на одной ноде, а команды SQL 2> на другой.

SQL 1>CREATE TABLE SEASONS

(

ID NUMBER NOT NULL

, DATEBEGIN DATE

, DATEEND DATE

, PARTICIP\_COUNT NUMBER

, CONSTRAINT SEASONS\_PK PRIMARY KEY

(

ID

)

);

SQL 1> COMMIT;

SQL 1> INSERT INTO SEASONS (ID, DATEBEGIN, DATEEND) VALUES (0, TO\_DATE('01-JAN-2010'), TO\_DATE('01-DEC-2010'));

SQL 1> INSERT INTO SEASONS (ID, DATEBEGIN, DATEEND) VALUES (1, TO\_DATE('01-JAN-2011'), TO\_DATE('01-DEC-2011'));

SQL 1> INSERT INTO SEASONS (ID, DATEBEGIN, DATEEND) VALUES (2, TO\_DATE('01-JAN-2012'), TO\_DATE('01-DEC-2012'));

SQL 1>COMMIT;

SQL 2> select \* from seasons;

 ID DATEBEGIN DATEEND PARTICIP\_COUNT

---------- --------- --------- --------------

 0 01-JAN-10 01-DEC-10

 1 01-JAN-11 01-DEC-11

 2 01-JAN-12 01-DEC-12

Заполненная таблица на одной ноде тут же доступна из другой ноды, что говорит о корректной настройке и работе RAC.

Выполним далее требуемые задачи.

* Создать нового пользователя "delightedbroodling".

SQL 1> CREATE USER delightedbroodling IDENTIFIED BY 214634;

User created.

SQL 1> GRANT CONNECT, RESOURCE TO delightedbroodling;

Grant succeeded.

* Закрыть все сессии на всех узлах кластера, запущенные от имени пользователя "delightedbroodling".

Для выполнения данного пункта залогинемся на второй ноде из-под нового пользователя.

$ sqlplus

SQL\*Plus: Release 11.2.0.1.0 Production on Sun Dec 20 18:20:54 2015

Copyright (c) 1982, 2009, Oracle. All rights reserved.

Enter user-name: delightedbroodling

Enter password:

Connected to:

Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production

With the Partitioning, Real Application Clusters, Automatic Storage Management, OLAP,

Data Mining and Real Application Testing options

SQL>

 Затем на первой ноде произведём поиск сессий данного юзера.

SQL 1> SELECT inst\_id, sid, serial# FROM gv$session WHERE username='DELIGHTEDBROODLING';

 INST\_ID SID SERIAL#

---------- ---------- ----------

 2 147 115

Зная номер инстанса, номер сессии и порядковый номер можно выполнить административную команду разрыва сессии.

SQL 1> ALTER SYSTEM KILL SESSION '147,115,@2';

ORA-00031: session marked for kill

SQL 1> SELECT status FROM gv$session WHERE username='DELIGHTEDBROODLING';

STATUS

--------

KILLED

 После этого все операции пользователя на второй ноде уже не будет доступны.

SQL 2> select \* from dual;

select \* from dual

\*

ERROR at line 1:

ORA-00028: your session has been killed

* Создать нового пользователя "exciteddrone".

SQL 1> CREATE USER exciteddrone IDENTIFIED BY 208145;

User created.

SQL 1> GRANT CONNECT, RESOURCE TO exciteddrone;

Grant succeeded.

* Закрыть все сессии на всех узлах кластера, запущенные от имени пользователя "delightedbroodling".

См. выше.

* Создать новую сессию от имени пользователя "exciteddrone".

Элементарный вход в систему инициализирует сессию. Поэтому можно просто зайти в систему из-под данного пользователя.

$ sqlplus

SQL\*Plus: Release 11.2.0.1.0 Production on Sun Dec 20 18:20:54 2015

Copyright (c) 1982, 2009, Oracle. All rights reserved.

Enter user-name: exciteddrone

Enter password:

Connected to:

Oracle Database 11g Enterprise Edition Release 11.2.0.1.0 - 64bit Production

With the Partitioning, Real Application Clusters, Automatic Storage Management, OLAP,

Data Mining and Real Application Testing options

SQL>

На следующем этапе произведём экстренную остановку одной из нод и проверим работоспособность системы в данном случае.

При нормальной работе система видит две ноды, в чём можно убедиться сделав запрос.

SQL 1> select inst\_name from v$active\_instances;

INST\_NAME

------------------------------------------------------------

rac1.localdomain:ractp1

rac2.localdomain:ractp2

Произведём экстренное отключение второй ноды – отключим виртуальную машину прямо во время работы.

После этого повторный запрос произведётся с некой задержкой (ожидание восстановления соединения, которое не произойдёт), после чего мы увидим, что система перераспределила всю работу на оставшуюся ноду.

SQL 1> select inst\_name from v$active\_instances;

INST\_NAME

------------------------------------------------------------

rac1.localdomain:ractp1

Проверим целостность данных запросом к базе данных:

SQL 1> select \* from seasons;

 ID DATEBEGIN DATEEND PARTICIP\_COUNT

---------- --------- --------- --------------

 0 01-JAN-10 01-DEC-10

 1 01-JAN-11 01-DEC-11

 2 01-JAN-12 01-DEC-12

Несмотря на потерю целой ноды данные по-прежнему активны и доступны для пользователя.

После этого включим вторую ноду произведём логин в базу данных. RAC на лету подхватывает информацию о доступности новой ноды и вновь перераспределяет ресурсы на обработку.

SQL 2> select inst\_name from v$active\_instances;

INST\_NAME

------------------------------------------------------------

rac1.localdomain:ractp1

rac2.localdomain:ractp2

Следующим этапом нам предстоит добавить и удалить сторонний vote disk. Посмотреть текущее состояние vote disk можно при помощи данной команды.

$ ./crsctl query css votedisk

## STATE File Universal Id File Name Disk group

-- ----- ----------------- --------- ---------

 1. ONLINE 6deac915f99c4fd9bf4b77654584500c (/dev/oracleasm/disks/DISK1) [DATA]

Создадим отдельный файл, представляющий собой vote disk.

# dd if=/dev/zero of=/u01/app/11.2.0/intriguedwraith bs=1M count=50

50+0 records in

50+0 records out

52428800 bytes (52 MB) copied, 0.163257 s, 321 MB/s

Двумя командами добавим дополнительный vote disk и удалим его.

$ ./crsctl add css votedisk /u01/app/11.2.0/intriguedwraith

Now formatting voting disk: /u01/app/11.2.0/intriguedwraith.

CRS-4603: Successful addition of voting disk /u01/app/11.2.0/intriguedwraith.

$ ./crsctl delete css votedisk 6deac915f99c4fd9bf4b77654584500c

CRS-4611: Successful deletion of voting disk 6deac915f99c4fd9bf4b77654584500c

Важно отметить, что в случае хранении всех данных внутри ASM нельзя подключить внешние vote disk как отдельные файлы – это мера предосторожности для сохранения целостности представления данных внутри ASM.

$ ./crsctl add css votedisk /u01/app/11.2.0/intriguedwraith

CRS-4258: Addition and deletion of voting files are not allowed because the voting files are on ASM

В случае работы с ASM следует создавать отдельные дисковые группы для этого.

Вывод:

В рамках данной работы мы познакомились с основами кластерного хранения данных на примере организации RAC. Мы провели полную конфигурацию нашей системы, а также проверили её работоспособность в ожидаемых и экстремальных условиях (потеря связи с одной из нод).