在 Solaris™ 容器中运行 Oracle 数据库的最佳做法

Ritu Kamboj 和 Fernando Castano

2005 年 9 月

内容

1 执行概要

本文档概述了 Solaris 10 操作系统中的 Solaris 容器，并包含在容器中运行非 RAC Oracle 数据库的准则。Oracle 9i R2 和 10g R1 数据库（RAC 和非 RAC）经过验证可在全局区域中运行。本文档重点介绍了如何在 Solaris 操作系统的容器中运行非 RAC Oracle 数据库，并详细说明了创建适用于部署 Oracle 数据库的非全局区域的过程。此外，本文档还指出了在 Solaris 操作系统的容器中运行非 RAC Oracle 数据库时的特殊注意事项。

对于本文档的其余部分，“Solaris 10 操作系统中的 Solaris 容器”称为“Solaris 容器”或简称为“容器”，并假设除非明确说明，否则容器始终与非全局区域相关联。另外，“非 RAC Oracle 数据库”简称为“Oracle 数据库”。

2 文档范围

Sun 和 Oracle 之间达成的新许可协议将设置了上限的 Solaris 10 容器视为硬分区（请参见参考资料 [6] 和 [7]）。本文档的范围是定义适用于运行 Oracle 数据库的 Solaris 10 操作系统容器。此外，本文档还指出了在容器中运行 Oracle 数据库的一些限制和特殊情况。
本文档未说明如何使用 Solaris 容器技术合并同一系统单独容器中的多个 Oracle 数据库实例。有关使用 Solaris 容器技术进行服务器合并的详细信息，请参见参考资料 [1] 和 [3]。

3 概述

本部分简要介绍了 Solaris 容器技术。此外，还介绍了 Solaris Zones 功能和 Solaris Resource Manager，它们是 Solaris 容器的两个主要组件（有关这些技术的详细信息，请参见参考资料 [2] 和 [4]）。

3.1 Solaris 容器

Solaris 容器旨在为应用程序提供一个完善、隔离且安全的运行环境。这种技术使用由软件定义的灵活界限，将应用程序组件彼此隔离起来。Solaris 容器旨在为应用程序所使用的资源提供精细控制，以使多个应用程序能够在单个服务器上运行，同时保持指定的服务级别。

Solaris 容器是一种管理结构，旨在为从中执行一组 Solaris 进程的环境提供统一的定义和管理模型。Solaris 容器使用 Solaris Resource Manager (SRM) 功能以及 Solaris Zones 来提供一个虚拟化环境，这种环境可以为应用程序工作负载提供固定的资源界限。

3.2 Solaris Zones 分区技术

Solaris Zones（Solaris 容器环境中的一个组件）是一种软件分区技术，它用于虚拟化操作系统服务，并为运行的应用程序提供隔离且安全的环境。Solaris Zones 非常适用于在单个服务器上合并多个应用程序的环境。

共有两种类型的区域：全局区域和非全局区域。底层操作系统（由系统硬件引导的 Solaris 实例）称为全局区域。每个系统中只有一个全局区域，它既是系统的缺省区域，又是用于系统范围管理控制的区域。全局区域管理员可以创建一个或多个非全局区域。创建非全局区域后，各个非全局区域管理员可以对这些区域进行管理，但他们的权限仅限于该非全局区域。

可以使用不同的根文件系统模型来创建以下两种类型的非全局区域：稀疏根区域和完全根区域。稀疏根区域模型通过以下方法优化对象共享：仅安装一部分根软件包，并使用只读回送文件系统获取对其他文件的访问。在此模型中，将 /lib、/platform、/sbin 和 /usr 目录缺省挂载为回送文件系统。此模型的优点是提高了性能，因为它可以有效地共享可执行文件和共享库，并大大减少了区域本身占用的磁盘空间量。完全根区域模型则最大限度地提高了可配置性，它将所需的软件包和选定的所有可选区域安装到区域的专用文件系统中。此模型的优点包括区域管理员可以自定义其区域文件系统布局，并且还可以任意添加未捆绑的软件包或第三方软件包。

Solaris Zones 提供了标准 Solaris 接口和应用程序环境。它们并不强制要求使用新的 ABI 或 API。通常情况下，无需将应用程序移植到 Solaris Zones。但是，在非全局区域中运行的应用程序需要注意非全局区域行为，尤其是：

- 非全局区域中运行的所有进程具有一组有限的权限，它们仅是全局区域中可用权限的一部分。如果进程所需的权限在非全局区域中不可用，则此进程可能无法运行，或者在某些情况下，进程无法达到最佳性能（例如，将 Oracle 配置为使用容器中的 DISM 时）。
- 每个非全局区域都有其自身的逻辑网络和回送接口。这些区域对上层流和逻辑接口之间的绑定进行了限制，以使流只能建立到同一区域中的逻辑接口的绑定。同样，只能将来自逻辑接口的数据包传递到此接口所在区域中的上层流。
- 非全局区域只能访问有限的一组设备。通常情况下，设备是系统中的共享资源。因此，在区域中设置了一些限制，以防危及系统安全。

3.3 Solaris Resource Manager

缺省情况下，Solaris 操作系统为系统中运行的所有工作负载提供了对所有系统资源的同等访问权限。Solaris Resource Manager 可以修改 Solaris 操作系统的这种缺省行为，它提供了一种控制资源使用的方法。

SRM 提供了以下功能：

- 对工作负载进行分类的方法，以使系统知道哪些进程属于给定的工作负载。
- 测量工作负载的功能，以评估工作负载实际使用的系统资源量。
- 控制工作负载的功能，以使工作负载不会相互影响，并获得所需的系统资源以满足预定义的服务级别协议。

SRM 提供了三种类型的工作负载控制机制：

- *约束机制*，Solaris 系统管理员可通过此机制限制允许工作负载使用的资源。
- *调度机制*，它指的是一些分配决策，用于满足负载较少或过高情况下所有不同工作负载的资源要求。
- *分区机制*，用于确保为给定的工作负载分配预定义的系统资源。

3.3.1 工作负载标识

项目：

项目是对工作负载进行标识和区分的一种方法。工作负载可以由几个属于不同组和用户的应用程序和进程组成。项目提供的标识机制用作工作负载的所有进程的*标记*。可以通过项目名称服务数据库在多个计算机之间共享此标识符。此数据库可能位于文件、NIS 或 LDAP 中，具体取决于 /etc/nsswitch.conf 文件中的项目数据库源的定义。资源控制机制使用分配给项目的属性为每个项目提供资源管理环境。

任务：

任务提供了对工作负载进行标识的另一种粒度级别。任务收集一组进程，并将其放入一个表示工作负载组件的易于管理的实体中。每次登录时都会创建一个属于项目的新任务，在登录会话过程中启动的所有进程都属于此任务。有些管理命令中已加入了项目和任务的概念，如 ps、pgrep、pkill、prstat 和 cron。

3.3.2 资源控制

可通过设置资源使用限制来控制工作负载的资源使用。这些限制可用于防止工作负载过度使用某一特定资源，而对其他工作负载造成不利的影响。Solaris Resource Manager 提供了一种资源控制功能，以便对资源使用加以限制。

每种资源控制都是由下面的三个值定义的：

- 权限级别
- 阈值

- 与特定阈值相关联的操作

权限级别表示修改资源所需的权限。它必须是以下三种类型之一：

- Basic（基本），进行调用的进程的所有者可以修改此级别
- Privileged（特权），只有特权（超级用户）调用者能够修改此级别
- System（系统），此级别在操作系统实例运行期间是固定的

资源控制的阈值构成了可以触发操作的执行点。在达到特定阈值时，将执行指定的操作。全局操作适用于系统中每种资源控制的资源控制值。局部操作是针对试图超出控制值的进程执行的。共有三种类型的局部操作：

- none（无）：当请求的资源量大于阈值时，不执行任何操作。
- deny（拒绝）：当请求的资源量大于阈值时，拒绝资源请求。
- signal（信号）：当超出资源控制值时，启用全局信号消息操作。

例如，task.max-lwp=(privileged, 10, deny) 通知资源控制功能拒绝 10 个以上的轻量进程访问该任务中的任何进程。

3.3.3 CPU 和内存管理

SRM 分别提供了*公平份额调度器 (Fair Share Scheduler, FSS)* 和*资源上限设置守护进程*功能，从而使最终用户能够控制系统中不同工作负载对可用 CPU 资源和物理内存的使用。

公平份额调度器

Solaris 操作系统中的缺省调度器为每个进程提供同等的 CPU 资源访问权限。但是，在同一系统中运行多个工作负载时，某个工作负载可能会独占 CPU 资源。公平份额调度器提供了一种机制，根据工作负载的重要性来设置 CPU 资源的访问优先级。

对于 FSS，工作负载的重要性是以系统管理员为表示工作负载的项目所分配的份额数表示的。份额定义了项目相对于其他项目的重要性。如果认为项目 A 的重要性是项目 B 的两倍，则分配给项目 A 的份额应该是项目 B 的两倍。

一定要注意，只有在存在 CPU 资源竞争的情况下，FSS 才会限制 CPU 使用。如果系统中只有一个活动项目，则它可以使用 100% 的系统 CPU 资源，而无论为其分配了多少份额。

资源上限设置守护进程

资源上限设置守护进程 (rcapd) 可用于控制定义了资源上限的项目所使用的物理内存量。对于配置了物理内存上限的项目，rcapd 守护进程反复对其内存使用情况进行抽样检查。抽样的时间间隔由管理员指定。当系统的物理内存使用量超出上限执行阈值并满足其他条件时，守护进程将采取措施以减少项目使用的内存量，并将其内存上限级别设置为等于或低于此上限。

请注意，rcapd 守护进程无法确定与其他进程之间共享的内存页面，也无法确定在相同进程中多次映射的内存页面。因此，如果项目使用 rcapd 限制物理内存使用量，建议不要在其中运行共享内存密集型应用程序（如 Oracle 数据库）。

4 Solaris 容器的 Oracle 许可证模型

目前，Oracle 将设置了上限的 Solaris 10 容器视为有权授予许可的实体，它称为硬分区。在 Solaris 10 环境中运行 Oracle 的 Oracle 客户现在可以只对设置了上限的 Solaris 容器中的 CPU 或核心授予许可，如下所述。

Oracle 许可策略将硬分区定义为“相当于自包含服务器的服务器的物理子集”（请参见参考资料 [2] 以了解详细信息）。以下示例说明如何使用 Solaris 10 操作系统

中的 Solaris 容器技术将 8 处理器系统划分为 3 处理器子系统。

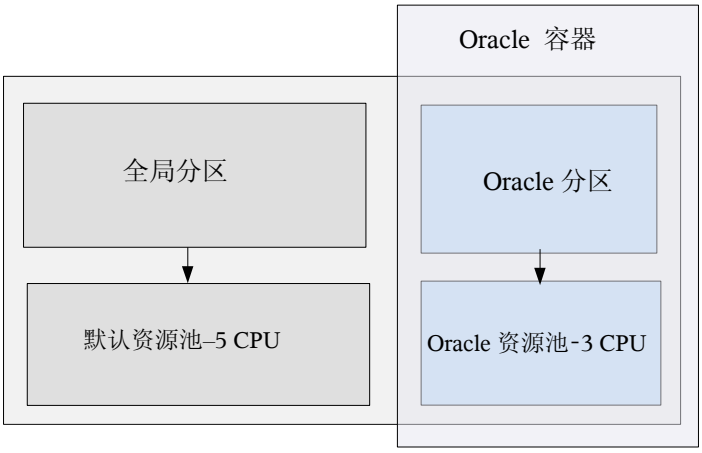

图 1：Solaris 10 操作系统中的 Oracle 容器示例

要创建符合 Oracle 所设置的许可要求的 Solaris 10 容器，Solaris 系统管理员需要创建具有所需数量的 CPU 或核心的资源池，并将一个区域绑定到该资源池上。许可证由该池中的 CPU 或核心数确定。

5 创建容器

本部分说明如何创建适合安装和运行 Oracle 的 Solaris 10 容器。这些说明为创建此类容器提供了一种简便的方法，附录 1 中给出的样例脚本均遵循这些说明。

5.1 要求

1. 确保放置容器根目录的文件系统至少具有 6 GB 的物理磁盘空间。这些空间足够创建容器和安装 Oracle。
2. 指定用于生成容器虚拟接口的物理接口。要查找全局容器中的可用物理接口，请执行以下命令：

   ```
   /usr/sbin/ifconfig -a
   ```

   通用接口的示例包括 ce0、bge0 或 hme0。
3. 获取容器的 IP 地址和主机名。此 IP 地址必须与在上一步中选定的物理接口所分配的 IP 地址属于同一子网。
4. 确保可以在全局容器中根据 /etc/nsswitch.conf 文件中使用的数据库解析容器 IP 地址的子网掩码。如果无法解析，则使用此 IP 地址所属子网的子网掩码更新全局

容器中的 /etc/netmasks 文件。

5. 确定为容器保留的 CPU 数。要找出缺省池中的可用 CPU 数，请执行 poolstat 命令。缺省池将指示可用 CPU 数。切记，缺省池必须始终至少有一个 CPU。

5.2 创建资源池

超级用户可以按照以下步骤在全局容器中创建资源池：

1. 使用以下命令启用池功能：

```
pooladm -e
```

2. 使用附录中提供的 pool_cmd_template.txt 文件作为模板来创建命令文件。将字符串 PSET_NAME、NUM_CPUS_MIN、NUM_CPUS_MAX 和 POOL_NAME 替换为相应的值。此命令文件包含一些指令，用于创建资源池和处理器集 (pset)，然后将 pset 与资源池相关联。

3. 如果缺省配置文件 /etc/pooladm.conf 不存在，可通过执行以下命令来创建该文件：

```
pooladm -s /etc/pooladm.conf
```

4. 通过执行以下命令来创建资源池：

```
poolcfg -f pool_commands.txt
```

其中，pool_commands.txt 是两步之前创建的文件。

5. 通过执行以下命令实例化对静态配置所做的更改：

```
pooladm -c
```

6. 通过执行以下命令来验证配置：

```
pooladm -n
```

5.3 创建非全局区域

创建资源池后，便可以创建非全局区域并将其绑定到资源池。就安装和运行 Oracle 而言，非全局区域可以使用完全根模型或稀疏根模型。我们建议使用稀疏根模型，除非它与特定要求有冲突。可按以下方式创建非全局 Solaris 区域：

1. 以超级用户身份创建一个目录，用于放置非全局区域的根目录（如 /export/home/myzone），然后将访问权限设置为 700。此目录的名称应该与区域名称（本示例中为 *myzone*）相匹配。

2. 除非添加了特殊指令，否则容器中的 /usr 目录将作为回送文件系统 (lofs)。这意味着，容器在只读模式下将全局容器中的 /usr 挂载到其自身文件系统树的 /usr 中。缺省情况下，Oracle 安装程序要求超级用户在 /usr/local 目录中创建文件。由于 /usr 在容器中是只读的，因此，我们将在

/usr/local 中创建一个专门的挂载点，以允许超级用户在容器中创建此类文件。请检查 /usr/local 目录在全局容器中是否存在，如果不存在，请创建该目录。

3. 在全局容器中创建一个目录，以便在容器中挂载 /usr/local。建议在 /opt/<ZONE_NAME>/local 中创建此目录。
4. 使用附录 1 中的 zone_cmd_template.txt 文件作为创建命令文件的模型来创建区域，并将其绑定到先前创建的资源池。将字符串 ZONE_DIR、ZONE_NAME、POOL_NAME、NET_IP 和 NET_PHYSICAL 替换为相应的值。请注意，此模板文件要求 /usr/local 的专门挂载点位于 /opt/<ZONE_NAME>/local 中。在此文件中，create 命令用于创建稀疏根。如果将此命令替换为 create -b，则会创建完全根。另外，在此文件中使用 set pool=POOL_NAME 命令将区域绑定到池上。另一种将池绑定到区域的方法是，在创建池和区域后执行 poolbind(1M) 命令。
5. 以超级用户身份执行以下命令以创建区域：

   ```
   zonecfg -z <ZONE_NAME> -f zone_commands.txt
   ```

   其中，zone_commands.txt 是在上一步中创建的文件。
6. 以超级用户身份执行以下命令以安装区域：

   ```
   zoneadm -z <ZONE_NAME> install
   ```
7. 使用以下命令引导区域：

   ```
   zoneadm -z <ZONE_NAME> boot
   ```
8. 使用以下命令完成容器配置：

   ```
   zlogin -C <ZONE_NAME>
   ```

6 特殊注意事项

在本部分中，我们指出了在容器中运行 Oracle 数据库的一些特殊注意事项。

6.1 容器中的设备

为了确保不损害安全性和隔离性，非全局区域中设置了某些与设备相关的限制。这些限制如下所示：

- 缺省情况下，在非全局区域中只能访问有限的一组设备（主要由伪设备组成），如 /dev/null、/dev/zero、/dev/poll、/dev/random 和 /dev/tcp。
- 公开系统数据的设备（如 dtrace、kmem 和 ksyms）在非全局区域中不可用。
- 缺省情况下，通过容器还无法访问物理设备。

区域管理员可以向非全局区域提供物理设备。管理员应负责确保上述操作不会危

及系统安全，这主要有以下两个原因：

1) 如果将物理设备放在多个区域中，可能会在区域之间创建转换通道。
2) 使用此类设备的全局区域应用程序可能会面临由非全局区域造成的数据泄露或数据损坏的风险。

全局区域管理员可以使用 zonecfg 的 add device 子命令，在非全局区域中加入额外的设备。例如，要将块设备 /dev/dsk/c1t1d0s0 添加到非全局区域中，管理员可以执行以下命令：

```
zonecfg -z my-zone
zonecfg:my-zone> add device
zonecfg:my-zone:device> set match=/dev/dsk/c1t1d0s0
zonecfg:my-zone:device> end
zonecfg:my-zone>exit
zoneadm -z my-zone reboot
```

可以在 match 命令中使用 /dev/dsk/c1t1d0*，以便将 /dev/dsk/c1t1d0 的所有分片添加到非全局区域中。可将此过程用于字符设备（也称原始设备）或任何其他种类的设备。

如果打算使用 Oracle 安装 CD 在非全局区域中安装 Oracle 数据库，您需要使 CD-ROM 设备在非全局区域中可见。为此，您可以从全局区域中使用回送命令挂载 /cdrom 目录，也可以导出物理设备（不建议使用这种方法）。有关如何从非全局区域中获取 CD-ROM 设备访问的详细信息，请参见参考资料 [8]。

6.2 容器中的文件系统

每个区域都有其自身的文件系统分层结构，它位于称为区域根目录的目录中。区域中的进程只能访问位于区域根目录下面的分层结构部分中的文件。

我们在下面列出了四种不同的方法，用于将文件系统从全局区域挂载到非全局区域中：

1. 在全局区域中创建一个文件系统，然后将其作为回送文件系统 (lofs) 挂载到非全局区域中。
    - 以全局区域管理员身份登录。
    - 在全局区域中创建一个文件系统。

        ```
        global# newfs /dev/rdsk/c1t0d0s0
        ```

    - 在全局区域中安装该文件系统。

        ```
        global#mount /dev/dsk/c1t0d0s0 /mystuff
        ```

    - 将 lofs 类型的文件系统添加到非全局区域中

```
global#zonecfg -z my-zone
zonecfg:my-zone> add fs
zonecfg:my-zone:fs>set dir=/usr/mystuff
zonecfg:my-zone:fs> set special=/mystuff
zonecfg:my-zone:fs>set type=lofs
zonecfg:my-zone:fs>end
```

2. 在全局区域中创建一个文件系统，然后将其作为 UFS 挂载到非全局区域中。

   - 以全局区域管理员身份登录。
   - 在全局区域中创建一个文件系统。

     ```
     global# newfs /dev/rdsk/c1t0d0s0
     ```

   - 将 ufs 类型的文件系统添加到非全局区域中。

     ```
     global# zonecfg -z my-zone
     zonecfg:my-zone>add fs
     zonecfg:my-zone:fs> set dir=/usr/mystuff
     zonecfg:my-zone:fs>set special=/dev/dsk/c1t0d0s0
     zonecfg:my-zone:fs>set raw=/dev/rdsk/c1t0d0s0
     zonecfg:my-zone:fs>set type=ufs
     zonecfg:my-zone:fs>end
     ```

3. 将设备从全局区域导出到非全局区域，然后从非全局区域中挂载此设备。

   - 以全局区域管理员身份登录。
   - 将原始设备导出到非全局区域。

     ```
     global # zonecfg -z my-zone
     zonecfg:my-zone> add device
     zonecfg:my-zone:device> set match=/dev/rdsk/c1t0d0s0
     zonecfg:my-zone:device>end
     zonecfg:my-zone>add device
     zonecfg:my-zone:device>set match=/dev/dsk/c1t0d0s0
     zonecfg:my-zone:device>end
     ```

   - 在非全局区域中以超级用户身份登录。
   - 在非全局区域中创建一个文件系统：

     ```
     my-zone# newfs /dev/rdsk/c1t0d0s0
     ```

   - 在非全局区域中挂载该文件系统：

     ```
     my-zone# mount /dev/dsk/c1t0d0s0 /usr/mystuff
     ```

4. 将 UFS 文件系统直接挂载到非全局区域的目录结构中。此处假定已向非全局区域提供了该设备。

   - 以非全局区域管理员身份登录。
   - 在非全局区域中挂载该设备：

     ```
     my-zone#mount /dev/dsk/c1t1d0s0  /usr/mystuff
     ```

6.3 使用恢复管理器进行备份

恢复管理器 (rman) 是 Oracle 数据库 8.0 和更高版本中提供的一个 Oracle 实用程序，用于备份、还原及恢复数据库文件。要在磁带上存储备份，RMAN 需要使用介质管理器。介质管理器是第三方软件，用于管理顺序存储介质（如磁带机）以备份和恢复数据库文件。虽然 rman 非常适用于非全局区域，但介质管理器可能没有经过验证可在非全局区域中使用。这可能意味着，rman 实用程序当前只能用于将数据库文件备份和还原到磁盘上。但是，可以将 rman 在磁盘上创建的备份从全局区域复制到磁带设备上。

6.4 卷管理

卷管理器通常是第三方产品，因此它们在容器中的可用性无法一概而论。到本文截稿时，还无法从容器中安装或管理卷管理器。目前，建议使用的方法是从全局容器中安装和管理卷管理器。在全局容器中创建设备、文件系统和卷后，可以使用 zonecfg 子命令向容器提供这些内容。

就 VERITAS 而言，容器中不支持以下功能：

- Admin ioctl
- 管理命令
- VERITAS Volume Manager 软件
- VERITAS Storage Migrator (VSM)
- VERITAS File System (VxFS)/VERITAS Federated Mapping Service (VxMS)
- Quick I/O 和 CQIO
- 群集文件系统

非全局区域中支持以下 VERITAS 功能：

- VERITAS 文件系统
- 从 lofs 文件系统的非全局区域访问全局区域中的 VxFS
- 从非全局区域中访问 ODM 文件
- 从非全局区域中执行并发的文件 I/O 操作

可以在非全局区域中访问 VERITAS 命令，但这些命令已进行了修改，以确保不会在非全局区域中执行。如果 VERITAS Volume Manager 命令检测到在非全局区域中执行了这些命令，将会显示以下错误消息：

```
VxVM command_xxx ERROR msg_id:  Please execute this operation in global zone.
```

类似地，也应该从容器的全局区域中安装和管理 Solaris Volume Manager (SVM)。按所需方式在全局区域中配置了存储后，可以向非全局区域提供这些元设备。

6.5 CPU 可视性

将区域绑定到资源池后，容器中的用户可能需要使用有关 CPU 可视性的虚拟化系统视图。在这些情况下，区域只能看到与它所绑定到的资源池相关联的 CPU。

Oracle 数据库应用程序主要使用 _SC_NPROCESSORS_ONLN 参数调用 pset_info(2) 和 sysconf(3c)，以获取其可用 CPU 数。Oracle 根据此数量确定内部资源大小，然后创建用于不同用途的线程（如并行查询和读取器数）。如果将区域绑定到具有关联 pset 的资源池，这些调用将返回所需的值（即资源池中的 CPU 数）。表 1 显示了 Solaris 操作系统中已修改的接口，以及在这种情况下返回所需值的接口。

接口	类型
p_online(2)	系统调用
processor_bind(2)	系统调用
processor_info(2)	系统调用
pset_list(2)	系统调用
pset_info(2)	系统调用
pset_getattr(2)	系统调用
pset_getloadavg(3c)	系统调用
getloadavg(3c)	系统调用
sysconf(3c)	系统调用
_SC_NPROCESSORS_CONF	sysconf(3c) arg
_SC_NPROCESSORS_ONLN	sysconf(3c) arg
pbind(1M)	命令
psrset(1M)	命令
psrinfo(1M)	命令
mpstat(1M)	命令
vmstat(1M)	命令
iostat(1M)	命令
sar(1M)	命令

表 1：具有容器识别功能且与 CPU 相关的 Solaris 10 接口列表

除了这些接口外，psrinfo(1M) 和 mpstat(1M) 等工具通常还使用某些内核统计数据 (*kstats*) 来检索系统的相关信息。这些 *kstats* 的所有使用者只能看到绑定到区域的池中的 *pset* 信息。

7 容器中不可用的 Oracle 功能

本部分介绍在容器的非全局区域中无法使用的一些 Oracle 数据库功能。

7.1 Oracle RAC

Oracle RAC 安装由一些节点、共享存储和专用互连组成。容器不能用作 Oracle RAC 节点，这主要有以下两个原因。

群集管理器：群集管理器是 RAC 安装中必须包含的软件组件之一。群集管理器负责将失败的系统与共享存储隔离开、避免数据损坏以及通过专用互连在群集节点之间实现通信。要将容器用作 RAC 节点，群集管理器必须能够对容器进行管理。到本文截稿时，还没有任何群集解决方案能够将非全局容器用作群集成员或节点。

Oracle RAC VIP：在容器中运行 Oracle RAC 的另一个限制是虚拟 IP。Oracle RAC 在每个节点中使用虚拟接口，将客户端连接分发到群集中的所有活动节点。如果从群集中删除了某个节点（节点 A），则群集中的某个其他活动成员（节点 B）将接管其虚拟 IP 地址，即生成一个额外的虚拟接口，它具有节点 A 所使用的 IP 地址。这样，节点 B 将处理发送到节点 A 的所有新连接，直至节点 A 重新加入到群集中。出于安全考虑，容器没有管理虚拟接口所需的权限。因此，Oracle RAC 使用的 VIP 机制与容器安全限制有冲突。

7.2 Solaris 动态锁定共享内存 (DISM)

本部分简要介绍了动态锁定共享内存 (Dynamic Intimate Shared Memory, DISM)，并说明了容器中不支持此功能的原因。

DISM 提供了可以动态调整大小的共享内存。使用 DISM 段的进程可以在运行过程中锁定和解锁部分内存段。通过执行此操作，应用程序可以根据系统物理内存的增加情况动态地进行调整，或者为系统物理内存的删除做准备。

缺省情况下，Oracle 在 Solaris 操作系统中使用锁定共享内存 (Intimate Shared Memory, ISM)，而不是标准 System V 共享内存。如果共享内存段成为 ISM 段，则会使用大页面对其进行映射，并锁定该段内存（即不能将其调出）。这会大大降低由于进程环境切换而产生的开销，从而改善了 Oracle 在有负载情况下的性能线性关系。有关 Oracle 和 DISM 的更多详细信息，请参见参考资料 [5]。

ISM 是自 Solaris 2.2 以后 Solaris 操作系统提供的一项功能，Solaris 2.4 和更高版本中缺省启用此功能。从 7.2.2 版开始，Oracle 缺省使用 ISM。在 Oracle 7.2.2 之前，通过在 init.ora 文件中添加 use_ism=true 指令，可以在 Oracle 中打开 ISM。Oracle 缺省使用 ISM 并建议使用此功能。容器中支持 ISM。

毋庸置疑，ISM 优于标准 System V 共享内存。但是，其缺点是无法调整 ISM 段的大小。要更改 ISM 数据库缓冲区高速缓存大小，您必须关闭并重新启动数据库。DISM 提供了可以动态调整大小的共享内存，从而消除了这一限制。自

Oracle 9i 开始，Oracle 数据库中支持 DISM。如果 init.ora 中的 sga_max_size 参数所设置的 SGA 最大值大于其组件（即 db_cache_size、shared_pool_size 和其他更小的 SGA 组件）之和，Oracle 将使用 DISM 而不是 ISM。

在 ISM 中，由内核对内存页面进行锁定和解锁。而在 DISM 中，由 Oracle 进程 ora_dism_<$ORACLE_SID> 完成内存页面的锁定和解锁操作。要锁定内存，进程需要 proc_lock_memory 权限。Solaris 非全局区域中没有提供此权限。虽然在非全局区域中调用 DISM 时将启动 Oracle，但它无法在 Solaris 非全局区域中锁定 SGA 内存。这可能会导致性能下降，因此**不要将 Oracle 配置为在非全局区域中使用 DISM**。

8 附录

8.1 附录 1：用于创建容器的脚本

本附录中介绍的脚本可用来创建适合安装和运行非 RAC Oracle 数据库实例的容器。这些脚本并不表示它们是创建容器的唯一方法。它们是作为样例代码提供的，应根据具体要求和限制对其进行修改。
这些脚本先创建资源池以及具有最小和最大 CPU 数（这两个值是由用户指定的）的处理器集 (pset) 资源。将在缺省配置文件 /etc/pooladm.conf 中配置这些新资源（请参见 pooladm(1M) 以了解详细信息）。创建 pset 后，它将与资源池相关联。接下来，将使用用户提供的根目录、IP 地址和物理接口创建一个稀疏根区域。将在 /opt/<zone_name>/local 中创建一个专门的 /usr/local 挂载点，以便于安装 Oracle，因为 /usr/local 是某些 Oracle 实用程序的缺省安装目录。创建该区域后，它将绑定到资源池上。绑定到资源池的区域组合将定义可在其中安装和执行 Oracle 的容器。该容器（以及在其中运行的所有进程）只能访问与资源池关联的 CPU。要使用这些脚本，请将所有这些文件保存在相同的目录中，然后执行以下步骤：

1) 使用以下变量的相应值编辑 setenv.sh 文件：

- ZONE_NAME：区域的主机名
- ZONE_DIR：区域的根目录
- NET_IP：区域的 IP 地址
- NET_PHYSICAL：将在其中创建区域虚拟接口的物理接口
- NUM_CPUS_MAX：pset 资源中的最大 CPU 数
- NUM_CPUS_MIN：pset 资源中的最小 CPU 数

2) 从全局容器中执行以下命令：

```
./create_container.sh
```

3) 从全局容器中执行下列命令以配置此容器：

```
zlogin -C <zone_name>
```

下面列出并说明了组成此脚本的文件。

8.1.1 README.txt

本文件说明在使用这些脚本时如何创建容器。

此外，它还给出了一些有关如何执行某些常见操作的提示，例如，为区域提供对原始设备的访问或删除资源池。

```
The scripts in this directory can be used to create a container
suitable for installing and running non-RAC instances of Oracle database.
These
scripts do not represent the only way in which you can create an appropriate
container for Oracle; depending on your requirements and constrains you can
modify these scripts to fit your needs.

1) creating a container for Oracle
These scripts will first create a resource pool and a processor set (pset)
resource with a minimum and maximum number of CPUs in it (both values
specified by the user).  These new resources will be configured in the
default configuration file /etc/pooladm.conf (see pooladm(1M) for details).
Once created, the pset will be associated with the resource pool.  Next,
a sparse root zone will be created with the root directory, IP and
interface provided by the user.  A special mount point for /usr/local will
be created in /opt/<zone_name>/local to facilitate the oracle installation,
since /usr/local is the default directory for the installation of some of
the oracle utilities.  Once created, the zone will be bound to the resource
pool. The combination of the zone bound to the resource pools will define
the container in which Oracle can be installed and used.  This non-global
container (and all processes running in it) will only have access to the
CPUs associated to the resource pool.  To use these scripts follow these
steps:


 a) edit the file setenv.sh with appropriate values for:
   - ZONE_NAME: hostname for the zone
   - ZONE_DIR: directory for the root directory of the zone
   - NET_IP:  IP for the zone
   - NET_PHYSICAL: physical interface in which the virtual interface for
         the zone will be created
   - NUM_CPUS_MAX: maximum number of CPUs in the pset resource
   - NUM_CPUS_MIN: minimum number of CPUs in the pset resource
 b) from the global container run ./create_container.sh
 c) Once the container has been created run "zlogin -C <zone_name>"
   from the global container to finish configuring the zone.


2) giving your container access to raw devices
If you need to give your container access to a raw device
follow this example once the container has been created
(these commands must be issued from the global container):

zonecfg -z my-zone
zonecfg:my-zone> add device
```

```
zonecfg:my-zone:device> set match=/dev/rdsk/c3t40d0s0
zonecfg:my-zone:device> end
zonecfg:my-zone> exit
zonecfg -z my-zone halt
zonecfg -z my-zone boot

3) giving your container access to a file system
If you need to give your container access to a file system
created in the global container follow this example once the non-global
container has been created:

global# newfs /dev/rdsk/c1t0d0s0
global# zonecfg -z my-zone
zoncfg:my-zone> add fs
zoncfg:my-zone> set dir=/usr/mystuff
zoncfg:my-zone> set special=/dev/dsk/c1t0d0s0
zoncfg:my-zone> set raw=/dev/rdsk/c1t0d0s0
zoncfg:my-zone> set type=ufs
zoncfg:my-zone> end
zonecfg -z my-zone halt
zonecfg -z my-zone boot

4) to remove pool resources previously created by operating
directly on the kernel (see poolcfg(1M) for details) use these commands:
poolcfg -d -c 'destroy pool my_pool'
poolcfg -d -c 'destroy pset my_pset'

5) to uninstall and delete a previously created zone use these commands:
zoneadm -z $ZONE_NAME halt
zoneadm -z $ZONE_NAME uninstall -F
zonecfg -z $ZONE_NAME delete -F
```

8.1.2 setenv.sh

在此文件中，用户将定义用于创建容器的参数。

```
#!/usr/bin/sh

#host name for the zone
ZONE_NAME=myzone
#directory where to place root dir for the zone
ZONE_DIR=/export/home
#IP for the zone (make sure netmask can be resolved for this IP according to
# the databases defined in nsswitch.conf)
NET_IP=129.146.182.199
#interface used by the zone
NET_PHYSICAL=bge0
#min and max CPUs for the pool bound to the zone
NUM_CPUS_MIN=1
NUM_CPUS_MAX=1


# do not make changes beyond this point
POOL_NAME=pool_$ZONE_NAME
PSET_NAME=ps_$ZONE_NAME
export ZONE_NAME ZONE_DIR NET_IP NET_PHYSICAL
export POOL_NAME PSET_NAME NUM_CPUS_MIN NUM_CPUS_MAX
```

8.1.3 zone_cmd_template.txt

此文件包含一个用于创建区域的命令模板集。在使用用户定义的值替换某些字符串后，将使用此文件来创建区域。

```
create
set zonepath=ZONE_DIR/ZONE_NAME
set autoboot=true
set pool=POOL_NAME
add net
set address=NET_IP
set physical=NET_PHYSICAL
end
add fs
set dir=/usr/local
set special=/opt/ZONE_NAME/local
set type=lofs
end
verify
commit
```

8.1.4 pool_cmd_template.txt

此文件包含一个命令模板集，用于创建资源池和 pset 资源，并将它们相关联。在使用用户定义的值替换某些字符串后，将使用此文件来创建区域。

```
create pset PSET_NAME ( uint pset.min = NUM_CPUS_MIN ; uint pset.max
=NUM_CPUS_MAX)
create pool POOL_NAME
associate pool POOL_NAME ( pset PSET_NAME )
```

8.1.5 create_zone_cmd.sh

此脚本使用 sed 实用程序创建一个用于创建区域的命令文件。它将替换区域命令模板文件中用户指定的参数。此脚本由 create_container.sh 调用。

```
#!/bin/sh
# Copyright (c) 2005 Sun Microsystems, Inc. All Rights Reserved.
#
# SAMPLE CODE
# SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT
# THE SUITABILITY OF THE SOFTWARE, EITHER EXPRESS
# OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
# IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS
# FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.
# SUN SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED
# BY LICENSEE AS A RESULT OF USING, MODIFYING OR
# DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
```

```
echo $ZONE_DIR > /tmp/ZP.$$
REG_ZONE_DIR=`sed 's/\//\\\\\\\//g' /tmp/ZP.$$`
rm -rf /tmp/ZP.$$ > /dev/null

sed -e "
/ZONE_DIR/ {
  s/ZONE_DIR/$REG_ZONE_DIR/
}
/ZONE_NAME/ {
  s/ZONE_NAME/$ZONE_NAME/
}
/NET_IP/ {
  s/NET_IP/$NET_IP/
}
/NET_PHYSICAL/ {
  s/NET_PHYSICAL/$NET_PHYSICAL/
}
/POOL_NAME/ {
  s/POOL_NAME/$POOL_NAME/
}
/./ {
  p
  d
}"
```

8.1.6 create_pool_cmd.sh

此脚本使用 sed 实用程序创建一个用于创建资源的命令文件。它将替换池命令模板文件中用户指定的参数。此脚本由 create_container.sh 调用。

```
#!/bin/sh
# Copyright (c) 2005 Sun Microsystems, Inc. All Rights Reserved.
#
# SAMPLE CODE
# SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT
# THE SUITABILITY OF THE SOFTWARE, EITHER EXPRESS
# OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
# IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS
# FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.
# SUN SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED
# BY LICENSEE AS A RESULT OF USING, MODIFYING OR
# DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.

sed -e "
/NUM_CPUS_MIN/ {
  s/NUM_CPUS_MIN/$NUM_CPUS_MIN/g
}
/NUM_CPUS_MAX/ {
  s/NUM_CPUS_MAX/$NUM_CPUS_MAX/g
}
/POOL_NAME/ {
  s/POOL_NAME/$POOL_NAME/
}
/PSET_NAME/ {
  s/PSET_NAME/$PSET_NAME/
}
/./ {
```

```
  p
  d
}"
```

8.1.7 create_container.sh

这是主脚本。它使用 setenv.sh 中指定的参数来创建容器。

```
#!/usr/bin/ksh
# Copyright (c) 2005 Sun Microsystems, Inc. All Rights Reserved.
#
# SAMPLE CODE
# SUN MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT
# THE SUITABILITY OF THE SOFTWARE, EITHER EXPRESS
# OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE
# IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS
# FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.
# SUN SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED
# BY LICENSEE AS A RESULT OF USING, MODIFYING OR
# DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.

# script to create a container to run oracle RDBMS.
# to use this script follow the instructions in the README.txt file
# located in this directory.

. ./setenv.sh

# 1)..............................  validate setenv.sh values

#zone path exists?
if [ ! -d $ZONE_DIR/$ZONE_NAME ]
then
 mkdir -p $ZONE_DIR/$ZONE_NAME
 if [ $? = 1 ]
 then
  echo ERROR: could not create root directory
  exit 1
 fi
fi
chmod 700  $ZONE_DIR/$ZONE_NAME

#zone already exists?
zonecfg -z $ZONE_NAME info > /tmp/z.$$ 2>&1
cat /tmp/z.$$ | grep "No such zone" > /dev/null 2>&1
if [ $? -eq 1 ]
then
 echo "ERROR: zone $ZONE_NAME already exists. IF you want to remove it do:"
 echo "use zoneadm -z $ZONE_NAME halt"
 echo "use zoneadm -z $ZONE_NAME uninstall -F"
 echo "use zonecfg -z $ZONE_NAME delete -F"
 exit 1
fi
rm -rf /tmp/z.$$ > /dev/null 2>&1

#pset already created?
pooladm -e
pooladm | grep pset | grep $PSET_NAME > /dev/null 2>&1
if [ $? -eq 0 ]
```

```
then
 echo "ERROR: pset $PSET_NAME already exists.  Please choose a different \
pset name "
exit 1
fi

#/usr/local directory exists?
if [ ! -d /usr/local ]
then
 mkdir /usr/local
fi

#special mnt point for /usr/local exists?
if [ ! -d /opt/$ZONE_NAME/local ]
then
 mkdir -p /opt/$ZONE_NAME/local
fi

# 2)............................... pool creation
./create_pool_cmd.sh < pool_cmd_template.txt > /tmp/pool_commands.txt
pooladm -e                        # enable facility
#check for default config file exists, if not there
#create one with active configuration
if [ ! -f /etc/pooladm.conf ]
then
 pooladm -s /etc/pooladm.conf
fi
poolcfg -f /tmp/pool_commands.txt       # configure
pooladm -n > /tmp/pool.out 2>&1         # validate
if [ ! $? -eq 0 ]
then
 echo ERROR: invalid pool configuration.  please see /tmp/pool.out
 exit 1
fi
#instantiate config at /etc/pooladm.conf
pooladm -c

# 3)...............................  zone creation
./create_zone_cmd.sh < zone_cmd_template.txt > /tmp/zone_commands.txt
zonecfg -z $ZONE_NAME -f /tmp/zone_commands.txt
echo $ZONE_NAME was configured with this information:
echo ------------------------------------------------------
zonecfg -z $ZONE_NAME info
echo ------------------------------------------------------
zoneadm -z $ZONE_NAME install
zoneadm -z $ZONE_NAME boot

echo "to finish configuring your container please run: zlogin -C $ZONE_NAME"
```

8.2 附录 2：设置 System V IPC 内核参数

在 Solaris 10 操作系统之前，System V IPC 资源（主要包括共享内存、消息队列和信号量）是在 /etc/system 文件中进行设置的。这种实现方式具有以下缺点：

- 依靠 /etc/system 作为管理机制，这意味着重新配置后需要重新引导。
- 在 /etc/system 中设置参数时，简单的输入错误就可能会导致出现很难找出原因的配置错误。

- 传统实现方式所使用的算法采用的是静态大小的数据结构。
- 要为某个用户分配额外的资源，您必须允许所有用户使用这些资源，否则，就无法进行分配。由于资源量始终是固定的，一个用户几乎无法防止其他用户执行其所需的分配。
- 无法很好地查看参数值。
- 某些可调参数的缺省值太小。

在 Solaris 10 操作系统中消除了所有这些限制。Solaris 10 操作系统中的 System V IPC 实现不再要求更改 /etc/system 文件。它使用具有以下优点的资源控制功能：

- 现在，无需更改 /etc/system 文件（在多数情况下，是对资源控制进行更改）即可安装并引导 Oracle 实例。
- 现在，可以针对每个进程或每个项目对 System V IPC 功能使用加以限制（具体取决于所限制的资源），而无需重新引导系统。
- 这些限制均不会直接影响分配。可以将这些限制设置得非常高，而不会对系统产生直接影响。（请注意，这样做将允许用户不受限制地分配资源，这会对系统产生影响。）
- 不再向管理员公开实现内部细节，因而简化了配置任务。
- 与先前的可调参数相比，资源控制更少，名称更详细直观。
- 可以使用通用资源控制接口来查看限制设置，如 prctl(1) 和 getrctl(2)。
- 共享内存是根据为每个项目分配的内存总量进行限制的，而不是根据为每个段分配的内存总量。这意味着，管理员可以为用户提供分配许多段和大段的权限，而无需为用户提供创建许多大段的权限。
- 由于资源控制是管理机制，因此可以使用 project(4) 永久保留该配置，并且可以通过网络进行设置。

在 Solaris 10 操作系统中进行了以下更改：

- 现在，消息头是动态分配的。以前，所有消息头都是在模块加载时分配的。
- 现在，信号量数组是动态分配的。以前，信号量数组是从 seminfo_semmns 大小的 vmem 区域中分配的，这意味着分配可能会由于分段而失败。
- 现在，信号量撤消结构是针对每个进程和每个信号量数组动态分配的。信号量撤消结构的数量不受限制，并且始终与其对应的信号量数组一样大。以前，每个进程的撤消结构数量有限，这些结构是在模块加载时分配的。另外，每个撤消结构的大小相同并且是固定的。进程可能无法分配撤消结构，或者进程的撤消结构可能已满。
- 现在，信号量撤消结构将其撤消值保存为带符号的整数，因此，信号量值不会由于太大而无法撤消。
- 以前，使用所有功能从固定大小的名称空间中分配对象，并且是在模块加载时分配的。现在，所有功能的名称空间可以调整大小，并随着需求的增加而增大。

由于进行了这些更改，已删除了以下相关参数。如果 Solaris 系统的 /etc/system 文件中包含这些参数，则会忽略这些参数（请参见表 2）。

参数名	*简要描述*
semsys:seminfo_semmns	系统中的最大 System V 信号量数
semsys:seminfo_semmnu	System V 信号量系统所支持的撤消结构总数
semsys:seminfo_semmap	信号量映射中的条目数
semsys:seminfo_semvmx	可以设置的信号量最大值
semsys:seminfo_semaem	撤消结构中可以设置的信号量最大值
semsys:seminfo_semusz	撤消结构大小
shmsys:shminfo_shmseg	每个进程的段数
shmsys:shminfo_shmmin	最小共享内存段大小
msgsys:msginfo_msgmap	消息映射中的条目数
msgsys:msginfo_msgssz	消息段大小
msgsys:msginfo_msgseg	最大消息段数
msgsys:msginfo_msgmax	System V 消息的最大大小

表 2：Solaris 10 操作系统中不再需要的系统参数

如上所述，已经删除了很多 /etc/system 参数，原因很简单，以后不再需要使用这些参数了。其余参数的缺省值更加合理，因而更多应用程序可以直接运行，而无需设置这些参数。

表 3 列出了其余 /etc/system 参数的缺省值。

资源控制	*过时的可调参数*	*旧缺省值*	*新缺省值*
process.max-msg-qbytes	msginfo_msgmnb	4096	65536
process.max-msg-messages	msginfo_msgtql	40	8192
process.max-sem-ops	seminfo_semopm	10	512
process.max-sem-nsems	seminfo_semmsl	25	512

project.max-shm-memory	shminfo_shmmax	0x800000	1/4 物理内存
project.max-shm-ids	shminfo_shmmni	100	128
project.max-msg-ids	msginfo_msgmni	50	128
project.max-sem-ids	seminfo_semmni	10	128

表 3：Solaris 10 操作系统中系统参数的缺省值

设置 Oracle 安装的 System V IPC 参数

表 4 列出了《Oracle Installation Guide》（Oracle 安装指南）中建议的 /etc/system 参数值以及相应的 Solaris 资源控制。

参数	*Oracle 建议值*	*在 Solaris 10 操作系统中是否为必需参数*	*资源控制*	*缺省值*
SEMMNI (semsys:seminfo_semmni)	100	是	project.max-sem-ids	128
SEMMNS (semsys:seminfo_semmns)	1024	否	N/A	N/A
SEMMSL (semsys:seminfo_semmsl)	256	是	project.max-sem-nsems	512
SHMMAX(shymsys:shminfo_shmmax)	4294967295	是	project.max-shm-memory	1/4 物理内存
SHMMIN (shmsys:shminfo_shmmin)	1	否	N/A	N/A
SHMMNI (shmsys:shminfo_shmmni	100	是	project.max-shm-ids	128
SHMSEG (shmsys:shminfo_shmseg)	10	否	N/A	N/A

表 4：运行 Oracle 10g 时系统参数的建议值

由于缺省值高于 Oracle 建议值，您可能只需要设置 project.max-shm-memory 资源控制。以下部分详细说明了使用资源控制设置特定值的过程。

使用资源控制命令设置 System V IPC 参数

可以使用 prctl 命令来查看和更改资源控制值。请使用 -n 选项调用 prctl 命令以显示某个资源控制的值。以下命令显示指定进程的 max-file-descriptor 资源控制的值：

```
prctl -n process.max-file-descriptor <pid>
```

以下命令更新 project group.staff 中的 project.cpu-shares 值：

```
prctl -n project.cpu-shares -v 10 -r -l project group.staff
```

9 参考资料

[1] *Consolidating Applications with Solaris 10 Containers*
（使用 Solaris 10 容器合并应用程序），Sun Microsystems，2004
http://www.sun.com/datacenter/consolidation/solaris10_whitepaper.pdf

[2] *Solaris 10 System Administrator Collection -- System Administration Guide: Solaris Containers-Resource Management and Solaris Zones*
（Solaris 10 系统管理员文档集 -- 系统管理指南：Solaris 容器 - 资源管理和 Solaris Zones），Sun Microsystems，2005
http://docs.sun.com/app/docs/doc/817-1592

[3] *Solaris Containers--What They Are and How to Use Them*
（Solaris 容器 -- 概念及用法），Menno Lageman，Sun BluePrints OnLine，2005
http://www.sun.com/blueprints/0505/819-2679.html

[4] BigAdmin 系统管理门户中的 Solaris Zones 部分，Sun Microsystems
http://www.sun.com/bigadmin/content/zones

[5] *Dynamic Reconfiguration and Oracle 9i Dynamically Resizable SGA*
（动态重新配置和 Oracle 9i 可动态调整大小的 SGA），Erik Vanden Meersch and Kristien Hens，Sun BluePrints OnLine，2004
http://www.sun.com/blueprints/0104/817-5209.pdf

[6] Oracle Pricing with Solaris 10 Containers（使用 Solaris 10 容器的 Oracle 定价）
http://www.sun.com/third-party/global/oracle/consolidation/solaris10.html

[7] Oracle 的分区文档
http://www.oracle.com/corporate/pricing/partitioning.pdf

[8] *Bringing Your Application Into the Zone*（将应用程序引入区域），
Paul Lovvik 和 Joseph Balenzano，Sun 开发者网络文章，2005
http://developers.sun.com/solaris/articles/application_in_zone.html

10 作者介绍

Ritu Kamboj 是 Sun Microsystems 市场开发工程小组的一名高级工程师。她在 Oracle 小组工作，负责帮助 Oracle 采用 Sun 技术和提高 Sun 硬件性能。

Fernando Castano 于 2000 年 10 月加入 Sun 公司。此后，他一直作为技术人员在市场开发工程部工作。目前，他在 Oracle 小组工作，负责帮助 Oracle 采用 Sun 技术和提高 Sun 硬件性能。