An Oracle White Paper

March 2010

Oracle Automatic Storage Management and Thin Reclamation

Disclaimer

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.

Executive Overview

Automatic Storage Management (ASM) is a feature of Oracle's database made available in Oracle Release 10g. ASM is a purpose built file system and volume manager integrated in Oracle's database. It dramatically reduces administrative overhead associated with managing storage. Storage costs, both in administrative overhead and capital expenses are growing concerns for most enterprises. Storage vendors have introduced an array of features to reduce the acquisition cost side of the equation. One such feature is *Thin Provisioning*.

Thin provisioning is a feature common to many storage arrays. Thin provisioning offers a different approach for provisioning storage where the storage array allocates capacity to the thin provisioned storage volume as application writes data over time, not upfront at time of storage volume creation. This approach is justified because for most enterprises, since written storage utilization for most applications operates between 20 and 45 percent. With Thin provisioning, written storage utilization can be increased to near 100 percent. As a result, IT users can significantly reduce the storage capacity that is required to support their Oracle databases. Oracle has actively supported storage vendors' thin provisioning feature in conjunction with a database feature known as "autoextend." Autoextend enables a file's size used for tablespace to grow when applications add new data to database tables.

Oracle is introducing new capability to ASM for extending support of thin provisioning. Previously, space inside a storage array supporting thin provisioning could only grow as a tablespace became larger. If a tablespace shrunk or even if an entire database were removed from an ASM disk group, space that had previously been allocated inside the array could not be freed for allocation to a different application. This new capability takes the form of an administrative ASM command that enables a storage array to detect unused space after it is freed by ASM and return that space to an unallocated status inside the storage array.

Background

Thin provisioning is a feature common to many storage arrays for optimizing utilization of available storage. Thin provisioning enables on-demand allocation rather than up-front allocation of physical storage. This storage feature reduces unused space and improves storage utilization. With thin provisioning, storage capacity utilization efficiency can be improved towards 100 percent, incurring little administrative overhead. Thin provisioning enables organizations to delay storage purchases and save on environmental costs.

Disk volumes inside storage arrays presented to an operating system are called LUNs. A LUN has a defined size measured in physical blocks; physical blocks are typically 512 bytes in length. The operating system determines the LUN size through a SCSI command or other storage interface standard. Historically, a storage array allocates the exact number of physical blocks on physical disks inside the storage array to match the number of blocks reported to the operating system as the LUN size. In effect, the allocation of physical blocks on LUNs presented to the operating system happened when the storage administrator created the LUN.

With thin provisioning, instead of static allocation of physical blocks, the allocation happens only when blocks are written to the array during use. Upon creation of a thin provisioned volume, the storage array communicates to the operating system the availability of a LUN of a specific size. However the actual physical size of the storage supporting the LUN is much smaller, equivalent approximately to the actual data written to the LUN. Thin provisioned LUNs begin with a minimum number of physically allocated blocks. The number of allocated blocks is less than the total that is reported to the operating system. A thin provisioned LUN has a current count and a maximum count of physical blocks. The latter is established by the storage administrator when the LUN is created. The former is largely determined by the amount of data written and is reported by standard storage array management tools.

The usage model is that as applications write to a LUN, the storage array automatically and transparently grows the LUN in the background up to the maximum size defined when the LUN was created. The benefit of thin provisioning is that it eliminates the reservation of disk space for every LUN, some of which will never grow to the reservation size or will only do so some time well into the future when disk capacity is less expensive. Ultimately, thin provisioning reduces the requirement for total capacity.

For performance reasons, when an application writes to a thin provisioned LUN leading to an allocation of physical storage blocks, the array allocates multiple physical blocks. The number of blocks a storage array allocates at any point varies by product, but it may be as small as 16 kilobytes to as large as 42 megabytes. Additionally, the allocation of storage need not be linear but can be sparse as storage is accessed.

Thin Provisioning and Oracle

The thin provisioning feature in storage arrays was first tested using ASM[1]. An ASM disk group was created using several thin provisioned LUNs. A database was created in the disk group and its primary tablespace used the database autoextend feature. Autoextend is to databases as thin provisioning is to storage. As additional space is needed, the database will automatically grow the tablespace in size. The file backing the tablespace in this test resided in an ASM disk group. When the database would write to newly allocated database blocks during an autoextend action, the storage array would automatically allocate physical storage to support the tablespace growth.

Deallocation of Thin Provisioned Storage

While thin provisioning works well for significantly improving storage utilization, it lacks the ability to free space that is deallocated or otherwise deleted at the application level. In a database example, after dropping a tablespace, a storage array is not able to detect the fact that the disk space used by the deleted tablespace is free and can be deallocated. From the ASM perspective, the disk space from the dropped tablespace is free, but that space is still allocated inside the storage array.

This is a common problem across most application sets and the storage industry set is about to address this shortcoming by extending the SCSI standard to incorporate a reclaim operation in the SCSI standard. The concept is that a new SCSI command would be used to communicate to the physical storage array the regions on a LUN that have been deallocated after an application deletes file space. When the array receives that command, the array can free the previously allocated physical storage.

[1] In 2004 Oracle and 3PAR jointly tested thin provisioning with ASM. As a result of this testing, Oracle concluded that using 3PAR thin provisioning in conjunction with ASM and the database autoextend feature yielded minimal performance impact. Since this initial development with 3Par, Oracle has actively supported all vendors implementing thin provisioning.

This requires support in the operating system at the device driver and file system level, as well as support by the storage array platform.

Most storage vendors fully support the approach of using SCSI commands to reclaim storage space as one of several approaches to space reclamation. 3PAR and others also provide an alternative approach - “zero detection." - especially useful when reclamation is needed with Operating Systems that don’t support SCSI command reclamation or with Operating Systems that are in the process of providing SCSI command-based reclamation in the future. With zero detection, the array firmware or dedicated hardware marks blocks written with zeros as de-allocated. Utilities such as Windows SDelete that write all zeros to deleted files and VMware's utilities that writes zeros to unallocated regions of a virtual disks benefit from this approach. The writing of zeros leads to the storage array deallocating the physical blocks of storage.

ASRU Utility

The strategy for reclaiming previously allocated storage in an ASM disk group is through a stand-alone utility. The utility, *asru (**A**SM **S**torage **R**eclamation **U**tility)*, is a PERL script that accepts the name of the disk group for which space should be reclaimed. When executed, it writes blocks of zeros to regions on ASM disks where space is currently unallocated. The writing of zeros is detected by the storage array as freeing of the storage. The *asru* utility operates in three phases:

Compaction Phase - In this phase, asru logically resizes the disks downward such that the amount of space in the disk group is at the allocated amount of file space in the disk group, plus a small amount for reserved capacity. The default value for the reserved amount is 25 percent; however, that is a settable option in the utility. The resize operation of the disks is logical to ASM and has no effect on the physical disks. The effect of the resize operation is that file data in the ASM disk group is compressed near the beginning of the disks. The utility uses the appropriate database V$ table to determine the current allocated size of the disk group. The next phase does not begin until the ASM rebalance for the disk group has completed and verified as complete. The rate of rebalance is controlled with the ASM Power Limit system setting.

Deallocation Phase - During this phase, asru writes zeros above the region of where the ASM disks have been resized. The *asru* script calls a subscript called zerofill that writes zeros to unused areas on the disks. Storage vendors that are unable to perform zero detection are free to implement their own subscript that deallocates storage as appropriate to their storage array.

Expansion Phase - In the final phase, all the ASM disks will be resized to their original size that was determined when asru was started. This resize operation is a logical resize of the disks with respect to ASM and does not result in a reorganization of file data in the disk group. This completes the operation and results in most of the unallocated storage being freed.

ASRU Utility Description

```
Usage: asru [--delta percent] [--overwrite] [-a <sysasm|sysdba>]
<dgname>
```

This script shrinks all the disks in the disk group to a smaller size, calls the zerofill script and resizes all the disks to their original sizes. The original sizes of all the disks are stored in a file in the specified ASM disk group. New size of a disk is the used space plus the given delta percent of the remaining free space.

After the successful resize, the zerofill script is invoked by looking in the current directory for a zerofill script. If not found, the user is prompted to enter the path to the zerofill script. The command is constructed by appending the arguments and the zerofill script is executed. After it completes, all the disks in the disk group are resized to their original sizes.

Recovery is performed based on the presence of the thin provisioned file in the disk group. If the file is present, it is more likely that the previous operation is incomplete. If the existing disks and disk names that are obtained from the thin provisioned file are the same the disk group state is recovered by resizing the disks to the sizes that are recorded the thin provisioned file. If the disks are not same, the thin provisioned file is overwritten if the overwrite is specified and exit if not.

Options description:

delta - percent of the free space in each disk that is retained

overwrite - overwrite the thin provisioned file if already present

<dgname> - name of the disk group to be made thin provision

a - connection type i.e. sysasm or sysdba

<dgname> - name of the disk group to be made thin provision

Conclusion

With enterprises deploying ever increasing volumes of storage, thin provisioning is an important feature for the industry. Thin provisioning not only reduces the amount of storage required for a particular organization, it reduces administrative overhead associated with storage provisioning. Oracle is committed to helping its customers maximize their investments and minimize administrative costs. ASM’s storage-reclaim utility is just one example of that commitment.

Recognition

The Oracle ASRU utility was developed by Oracle in collaboration with 3PAR, a provider of Utility Storage. Many thanks go to the joint Oracle and 3PAR team for collaborative development and test efforts.

Oracle Automatic Storage Management and
Thin Reclamation
March 2010
Jim Williams

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

Oracle is committed to developing practices and products that help protect the environment

Copyright © 2010, Oracle and/or its affiliates. All rights reserved. This document is provided for information purposes only and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

0109