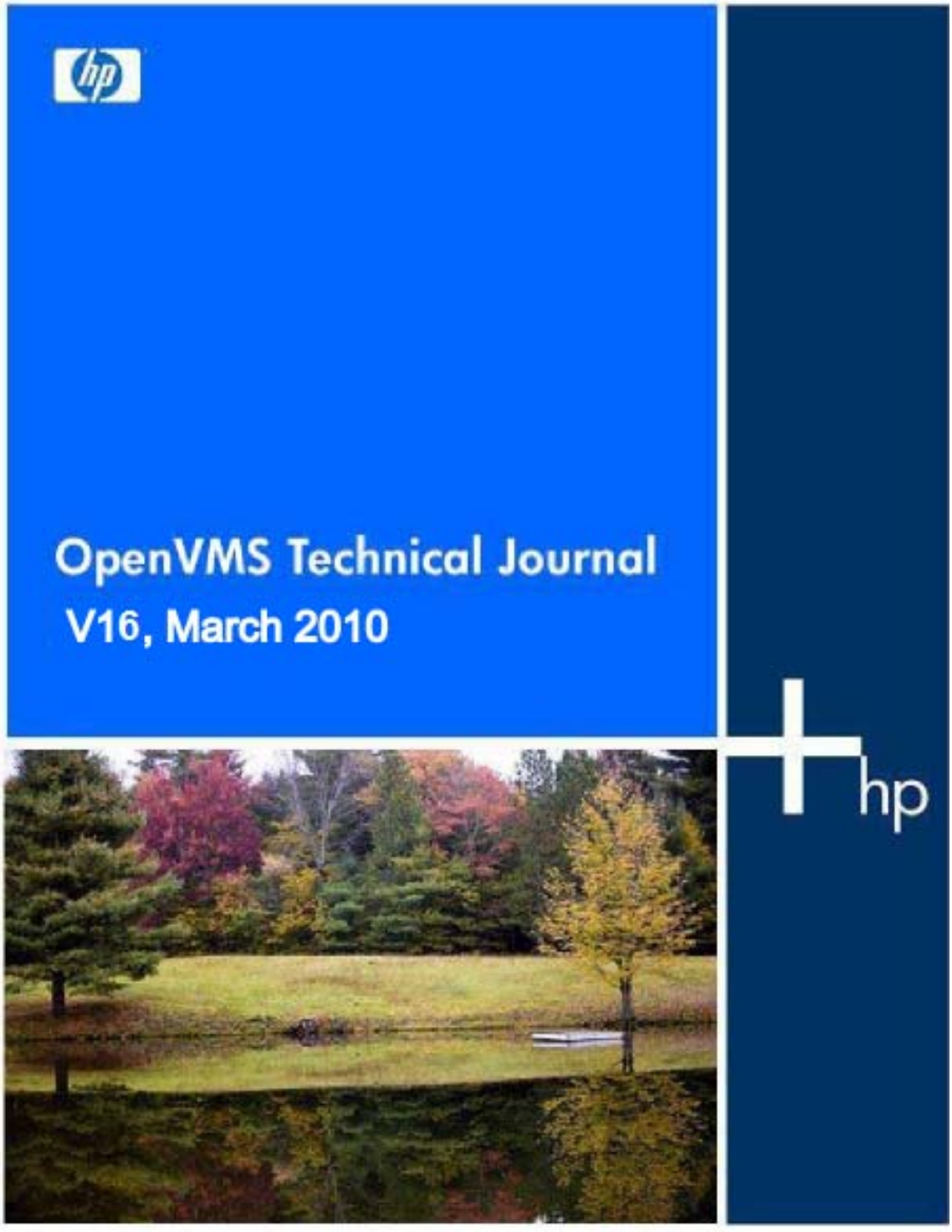

OpenVMS Technical Journal
V16, March 2010
hp

Table of Contents

OpenVMS Technical Journal V16

Archive Backup System and Data Protector: Leading OpenVMS Backup Solutions

Deepa R Shenoy, OpenVMS Engineering

Introduction

Backup strategy is a plan that aids the user to recognize the backup solution most suitable to the site. There is no particular backup strategy that can be termed as a solution for all the customer sites. Therefore, according to the site requirement, a backup strategy should be framed which would aid in choosing the best suitable backup solution.

Various Enterprise backup solutions are available for OpenVMS platform catering to wide spectrum of customer requirements/needs. HP Archive Backup System (ABS) is a backup software solution which is suitable for OpenVMS prominent environment. Whereas, HP Data Protector (DP) is a versatile product for the needs of today's heterogeneous enterprise environments including OpenVMS.

This paper will delve upon some of the useful features from both the products. The document does not compare the products to prove one solution better than the other, but merely lists the features of ABS and DP. The user needs to decide which solution best suits their environment and this feature listing may aid the user with the decision.

Product introduction

Archive Backup System

ABS is primarily developed for OpenVMS Operating System users. Apart from Open VMS files, ABS can be used to backup and restore TRU64 UNIX and Windows files. ABS supports HP devices that are qualified by OpenVMS Operating System. It also supports SUN libraries with the help of another product DCSC (Digital Cartridge Server Component). ABS manages small environments and can also be scaled to manage backups in enterprise environments.

The latest major version release of ABS is Version 4.5 (1200). After this, another patch or remedial release of ABS V4.5 (1201) was done which contains fixes to some of the problems reported in ABS. For more information about the latest release of ABS, see the ABS website: http://h71000.www7.hp.com/openvms/storage/abspage.html

Data Protector (DP)

Data Protector is designed to backup or restore data for heterogeneous enterprise environment. DP supports backup of a multitude of file systems available in the market which includes OpenVMS files backup. DP works with both HP and non-HP hardware (Tape Libraries) available. DP offers easy central management via GUI, that is, from a single system; all DP client systems can be accessed.

The latest version of DP is A.06.11. For more information on this DP release, see the DP Documents Page.

Product components

Archive Backup System

Following are some of the ABS components listed below:

1. *Media and Device Management System* (MDMS) is an integrated component of ABS. MDMS performs several functions for ABS:
   - Database Services: All ABS and MDMS objects are maintained by MDMS.
   - User Interface: MDMS manages both the Command-line Interface (CLI) and Graphical User Interface (GUI).
   - Security Service: MDMS manages access rights/privileges for both ABS and MDMS objects.
   - Media Management Services: MDMS takes care of Media Management for ABS. MDMS specifically supports a set of objects (termed as MDMS objects) for media management.
   - For Network backups, MDMS works in conjunction with RDF (remote Device Facility). RDF works only with Decnet and Decnet over IP. RDF does not work with TCPIP.

2. A *Catalog* is a set of files created to preserve the history information of backups done using ABS. Catalogs contain the records of data saved using ABS. Those records enable you to locate and restore data that was saved using ABS. ABS refers to the catalog for information during data restoration; this process is termed as "Catalog Lookup" in ABS.

3. *ABS objects* define physical locations of backed-up data on tape/disk, the criteria under which save and restore requests are performed that is the time and schedule of backups/restores.
   MDMS Objects define physical resources such as node, jukebox, drives, tape (cartridge/volume). It also defines logical resources such as media type, pool, magazine, and group.

4. *MDMS Domain* – Each OpenVMS node participating runs a generic process called MDMS$SERVER. A MDMS$SERVER process which has access to the database is termed as the *DB server*. Any other MDMS$SERVER process is served by the DBServer in the same MDMS domain, this is termed as MDMS client. All nodes communicating with the same database server belong to the same MDMS domain. Each MDMS domain has its own database. Typically you have only one MDMS domain in your network.

5. A *backup agent* is the utility that performs the actual data movement operation.
   OpenVMS systems - the backup agents are the OpenVMS BACKUP Utility,
   RDB backup - backup agent is RMU Backup Utility UNIX and Windows clients – backup agent is GTAR (tape archiver).

For more information on each of these components, see the ABS Guide to Operations:
http://h71000.www7.hp.com/doc/abs044_opr_gd.pdf

Data Protector (DP)

Following are some of the DP components explained below:

1. *DP Cell*: *Cell Manager* is a central control point where the Core Data Protector software is installed. The *Cell Manager* allows you to add systems to be backed up; these systems become DP Client systems to the *Cell Manager* and are considered a part of the Cell.

2. DP Client systems must have *Disk Agents* (or Backup Agents) installed. The Disk Agent reads or writes data from a disk on the system and sends or receives data from a *Media Agent*. The Disk Agent is also installed on the *Cell Manager*, thus allowing you to back up data on the *Cell Manager*, the DP configuration, and the IDB.

3. A *backup device* (Tape devices) can be connected to either the *Cell Manager* or any other client system. A system in the CELL which has a Backup device must have the Media Agent installed. The *Media Agent* reads or writes data from media on the backup device and sends or receives data from the disk agent.

4. The *IDB (DP Internal Database)* is a database located on the *Cell Manager*, which maintains history information regarding what data is backed up, on which media it resides, the result of backup, restore, object copy, object consolidation, object verification, and media management sessions, and what devices and libraries are configured IDB.

5. The *Manager of Managers* (MoM) allows a user to manage large enterprises with multiple cells. MoM is a single control point from which many cells can be managed. MoM facilitates sharing of devices between CELLs and aids in report generation starting from a single cell basis or for the entire enterprise (all CELLS under MoM).

Product installation

ABS Installation on OpenVMS

A single kit is provided for ABS installation on both OpenVMS Alpha and OpenVMS Integrity server Operating System. The ABS installation is done with the help of VMSINSTAL utility.

ABS needs to be manually installed on all OpenVMS systems identified as part of the backup strategy. Installation of ABS OpenVMS server or client software is determined by the OpenVMS node name that you enter during the installation procedure. There is no separate kit provided for an ABS OpenVMS client.

ABS allows a user to install the product in a standard mode, where ABS decides the host disk and product components to be installed. It also allows user to control the installation in the custom mode. The user can decide the disk, where ABS product must be installed. Along with this, the user can also choose the optional ABS components to be installed such as RDF, MDMS GUI, NT kits and so on. For information about ABS installation instructions, see the ABS Guide to Installation: http://h71000.www7.hp.com/doc/abs044_ins_gd.pdf

ABS Installation on different OS

ABS GTAR component needs to be installed on TRU64 UNIX systems and on Windows an ABS service needs to be installed and started up so that ABS can include the systems as clients. Apart from this, the Windows and TRU64 UNIX clients will need to be authorized on the OpenVMS system. For information about ABS installation instructions, in the ABS Guide to Installation:

http://h71000.www7.hp.com/doc/abs044_ins_gd.pdf

DP Installation on OpenVMS

The cell manager can be installed on the Windows, HP-UX, Solaris, or Linux platform. However, on an OpenVMS system, only a DP client may be installed. This paper discusses about the DP installation on OpenVMS. For more information, see the DP Installation Guide:

http://bizsupport2.austin.hp.com/bc/docs/support/SupportManual/c01631236/c01631236.pdf

DP OpenVMS client software needs to be manually installed on all OpenVMS systems identified as part of the backup strategy. You can install the Data Protector Disk Agent, General Media Agent, and the User Interface (command-line interface only) on systems running OpenVMS Alpha V7.3-2 or OpenVMS Integrity servers V8.2-1. You can also install the Oracle Integration component on systems running OpenVMS Alpha V7.3-2 or later. DP provides a PCSI file which needs to be installed manually on the OpenVMS system. The DP installation is done with the help of PCSI utility (PRODUCT INSTALL). The product can only be installed on the system disk in "SYS$COMMON:[OMNI]".

DP allows the user to install the product in a default mode, where all the components get installed by default. DP allows the user to control the installation in the Custom mode. The user decides the DP components that should be installed on that OpenVMS system. A step by step procedure is available in the DP Installation Guide under the section "Local installation of HP OpenVMS clients".

DP Installation on different OS

The Installation Server (IS) is a separate system or a Cell Manager component that contains the Data Protector Software repository used for remote client installations. This Data Protector feature greatly simplifies the software installation process for remote clients. Installation Server cannot be used to install DP on OpenVMS systems. Installation Server is supported for UNIX and Windows Operating Systems. Client software can be installed remotely on HP-UX, Solaris, Sinix, Linux, AIX, and other supported UNIX operating systems from an installation server for UNIX. Client software can be distributed to any Windows system, except Windows XP HE, from an installation server for Windows.

Types of backup

Full/Incremental backups

A full backup saves all the files selected for backup in a file system.

During a full backup, ABS performs an image backup on OpenVMS file system. There is no extra step required to make the system disk bootable during a restore.

On OpenVMS, DP has no equivalent functionality like the native OpenVMS image backup using BACKUP/IMAGE or BACKUP/PHYSICAL. To make a restored copy of an OpenVMS system disk

bootable, the OpenVMS WRITEBOOT utility has to be used to write a boot block onto the restored disk.

For ABS and DP, an incremental backup saves only those files that have changed since the last full or incremental backup.

Synthetic/Virtual full backup

This concept is unique to DP and ABS does not support these backup types. A synthetic full backup data set is consolidated from a full backup (residing on disk/tape) and incremental backups (residing on disk). This eliminates the need for the backup server to be involved during the backup hence speeding up the backup time.

If all the backups, full and incremental, are written to the same file library that uses distributed file media format, virtual full backups may be used. This solution uses pointers to consolidate data rather than copy the data. As a result, the consolidation takes less time and avoids unnecessary duplication of data.

DP does not support zero downtime backup (ZDB) or split mirror backups on OpenVMS.

Backup to disk

The resultant data-set of a backup operation needs to be stored on a separate medium that can be later transported to a different location or maintained as a near-line option. There are two types of media that the backup solutions use, they are disk medium (generally used as near-line option) and tape medium (best medium for maintaining offline data backups).

Both ABS and DP supports backups to disk. The user can specify the disk on which the data is to be stored. But ABS does not support disk-to-disk backups. DP, on the other hand, supports disk-to-disk backups. These disk-to-disk backups include:

- Disk Staging: Data from backup server is intermediately done to a STAGED DISK and later copied to a Tape device.
- Synthetic Full backup and Virtual Full Backups are examples of disk to disk backups.

To aid disk-to-disk backups, DP has the following disk-based devices:

- Standalone file device - It consists of a single slot to which data can be backed up.
- File jukebox device - The file jukebox device consists of multiple slots to which data can be backed up. Each slot in the file jukebox device has a maximum capacity of 2 TB. This requires a 2-stages configuration.
- File library device - The file library device has multiple slots called file depots to which data can be backed up. Each file depot has a maximum capacity of up to 2 TB. This requires a single stage configuration.

Backup to tape

The traditional and highly used medium is tape or cartridge. Tapes are used as they are reliable, cheaper than disk and can be easily transported from one place to another.

Data format on Tape

ABS and DP both support backup to tapes. ABS stores backed up data on a tape in a SAVESET format which is understandable to native VMS BACKUP utility. DP stores data on a tape in its own format understandable only to DP.

Media and device management

Tapes are a highly used medium for backups and tapes require media management to be done. Both products offer excellent media management facilities. These media managers provide common functionalities such as:

- Media pools: Logically grouping media to enable user to categorize media efficiently.
- Media details: Maintaining useful information about the media at all times, especially, expiration time of data on medium, availability of media etc.
- Media rotation policies: Automatically rotating tapes without need for manual intervention.
- Providing barcode support on large Libraries, and so on.

There are also other functionalities of media management where the products differ; some of them are explained below:

1. **Device streaming**

   If the rate at which data is written by the drive to the tape is less than or equal to the rate at which data is being provided to the device by the application, then the device is said to be 'streaming'. Device streaming is also dependent on other factors, such as, network load and the block size of the data written to the device. In DP, the device streaming is accomplished by starting multiple Disk Agents for each Media Agent that writes data to the device. If properly configured, this setup leads to increased backup performance.

   ABS does not include a device streaming mechanism.

2. **Load balancing**

   In both ABS and DP, multiple devices might be used for backup so that data is backed up in parallel to multiple devices (drives/media).

   But DP also supports load balancing that is, DP automatically balances the load or usage of devices for evenly using the devices. Load balancing provides optimum performance:

   - When large number of objects needs to be backed up
   - A library with several drives are available
   - Good network connection is available

   ABS does not take up the responsibility of load balancing.

3. **Media condition maintenance**

   When ABS/MDMS encounters an error with media, it automatically avoids the usage of that media for further backups. The user can later query the MDMS database to know the media which are not in use due to media errors. The user now has to manually check the problem with media and fix the errors. MDMS does not explicitly maintain the media condition based on the usage or age of media.

DP calculates the state of the media using "Media Condition Factors". In DP, media is set to have any one of the three states: GOOD, FAIR, OR POOR. The state is calculated on the basis of:

- Number of Overwrites - Once the medium has more than the threshold number of overwrites, it is marked as poor.
- Media Age - The age of a medium is calculated as the number of months that have elapsed since you formatted, or initialized, the medium. Once a medium is older than the threshold number of months, it is marked as poor.
- Device Errors - If a device fails during a backup, the medium used for the backup in this device is marked as poor.

Data Protector fully supports TapeAlert 2.0 (a device monitoring utility that provides error/warning alerts and suggest a course of action), as long as the connected device also provides this functionality.

4. Cleaning tape handling

In ABS, the cleaning tapes have to be manually loaded, whereas DP provides automatic cleaning for most devices using a cleaning tape. This medium will be used automatically by DP if a dirty drive event from the device is detected. For devices without a cleaning tape, dirty drive detection will cause a cleaning request to be displayed on the session monitor window. The operator must clean the device manually.

5. Tape size

DP is capable of displaying the approximate size of tape that remains to be filled. ABS does not maintain any information on the size of tape.

6. Virtual tape libraries (VTL)

The Backup window is further reduced and managing the backups has become easier and faster with VTLs. ABS supports HP VLS (Virtual Library System) and HP D2D (Disk 2 Disk device) for OpenVMS files backup but DP supports only HP VLS for OpenVMS file systems.

The latest technology that these Virtual Tape Libraries have come up with is called "deduplication". Currently (as on June 2010) VLS does not support deduplication of OpenVMS file systems. But D2D device supports deduplication of OpenVMS file system. ABS works well with deduplication of OpenVMS files along with the D2D device.

Disaster situation handling

Natural disaster

When a natural disaster occurs at the business data centre site, a remote data centre site needs to be setup immediately for business continuity. A disaster recovery from the last backed up data of business data centre needs to be performed at this stage.

Disaster Recovery of an OpenVMS system is an uncomplicated procedure in ABS and requires:

1. The image backup of the system disk on OpenVMS node.
2. Backup of important information such as, ABS Catalog files and MDMS database files.

Image backups create bootable disk at the time of restore. Image backups also automatically defragment the disk when a restore is attempted, which mostly results in improved performance. After the restore is complete, ABS and MDMS files are restored to make sure that the ABS/MDMS database is consistent and exact state is restored.

For more information, see Section 7 in the *ABS Guide to Operations*:

http://h71000.www7.hp.com/doc/abs044_opr_gd.pdf

Disaster Recovery of an OpenVMS system in DP requires:

1. A complete restore of files on disk (disk will be non-bootable).
2. The OpenVMS system needs to be booted from a different disk (another bootable disk).
3. With the help of WRITEBOOT, write the boot block on the disk to make the disk "bootable".

User error

An uncommon situation where a user has accidentally deleted the history information (Catalog of ABS/ IDB of DP) of a backup solution.

For ABS, the following prevention and remedy techniques can be used:

Prevention: ABS recommends you to always backup the catalogs and also set appropriate VMS ACLs and MDMS ACLs to protect catalog data.

Remedy: ABS provides a mechanism to rebuild lost catalogs with a method called "Cataloging Existing Saveset". Here the Customer can refer the backup logs and find out the tape name where the data is placed. ABS will rebuild the history data for all the SAVESETS present on this tape. ABS does not support this mechanism for encrypted SAVESETS.

For DP, the following prevention and remedy techniques can be used:

Prevention: The following features allow and restrict access to Data Protector Cells

- Data Protector user accounts
- Data Protector user groups
- Data Protector user rights

Remedy: DP provides a mechanism to rebuild the history information in IDB by a technique named "Importing catalog information from media".

Application server node crash

A common situation seen is when the Application server node crashes, there are several client nodes waiting to get serviced by the Application server. In ABS, a user can nominate more than one node (all nodes must have access to the DB disk to be the application Server (MDMS DB server). At a point in time, one node can act as application server. If the current Application Server (DB server) node crashes, automatically ABS takes care of the situation and fails over to the next nominated node in the list. When the crashed node comes up, it takes up the role of a client system getting serviced by the new Application Server.

In DP, all Cell Manager Operations are always available since Data Protector Services are defined as cluster resources within the cluster and are automatically restarted when a failover occurs.

Media hardware error

A common situation where the backup application has previously stored data on a tape, while using the tape for a subsequent backup, a media hardware error is encountered.

If the previously stored data is critical data and needs to be restored, ABS allows a user to restore data stored on tape till the point where tape allows access (until the failure point).

DP maintains media state and places the tape in POOR state after it experiences problems with the hardware. DP does not restore any data from media in POOR state.

Product features compared

Sl. No.	Features	Archive Backup System (ABS)	Data Protector (DP)
1	GUI	GUI on Windows is provided.	GUI on Windows, HP-UX, SOLARIS, UNIX platforms is provided.
2	CLI	ABS/MDMS CLI on OVMS provided	DP CLI on OVMS provided
3	Backup technique	Uses the native BACKUP Utility to perform backups	Issues QIO call over a network to backup data
4	Archiving (After Backup, the original data on disk is deleted)	Supported	Not supported
5	Data specification details	Include Specification (VMS style File specification is supported by ABS)	Backup Specification (Any file specifications that are passed to the CLI must conform to a UNIX-style syntax: /disk/directory1/directory2/filename.ext.n)
6	Network Backups	ABS supports Network Backups via RDF (Remote Device facility). But RDF works with Decnet and Decnet over IP only.	DP supports network backups (TCPIP).
7	Frequency of Backups	User can choose from a predefined frequency sets or define a desired custom frequency.	User can choose from a predefined frequency sets or define a desired custom frequency.

8	Schedulers	ABS supports usage of 1. MDMS internal scheduler 2. External scheduler (CA Scheduler)	DP Scheduler
9	Track data backed-up using	Catalogs	IDB (Resides on Cell Manager)
10	Procedure for cleaning up the above mentioned catalog/IDB	Catalog cleanup (configurable)	Purging policies (configurable)
11	Centralized repository (catalog/IDB)	Catalogs are generally node-specific and not centralized. But this is configurable.	IDB is centralized and optimized.
12	Maximum size of catalogs/IDB	Generally if the catalog size grows beyond 5 GB, ABS performance problems may be seen.	Generally if the IDB size grows beyond 16 GB, DP performance problems may be seen.
13	Catalog Segment on Tape	ABS does not store a Catalog segment on Tape.	DP stores a Catalog segment on tape with every data segment.
14	RDB support for OpenVMS	RDB V7.2 supported	Not qualified to work with RDB
15	Oracle support for OpenVMS	Oracle 10gR2 supported	Oracle 10gR2 supported
16	Hardware Encryption for OpenVMS files	ABS V4.5 also has been qualified with the latest MSL LTO-4 encryption kit that provides key management functionality for LTO-4 drives in an MSL G3 library.	Hardware encryption is not supported for OpenVMS files. For Oracle integration backup/restore AES algorithm (256 bit) for encryption is supported.
17	Software Encryption for OpenVMS files	Software Encryption feature was introduced in ABS starting from ABS V4.5 and ABS supplies its own key management facility.	Software Encryption of OpenVMS files backup is currently not available on DP.
18	Compression of OpenVMS Files	This feature is currently not available	With DP, compression of OpenVMS files is achievable.
19	Tape to tape copy	This feature is currently not available	With DP, tape to tape copy is supported which in turn helps in tape consolidation. I.e. copying discrete data on different tapes to a single tape.

20	ODS 2 and ODS 5 file system Support	Both are supported	Both are supported
21	Network	ABS can be configured to work with DECNET or TCPIP or both.	DP works with TCPIP alone for OpenVMS Clients
22	Multiple NIC card support	ABS does not support Multiple NIC card configuration.	DP supports a Multiple NIC card configuration.
23	Hardware Support details	Link for Device Matrix	Link for Device Support Matrix
24	Hard Links On Open VMS	ABS takes care of the hard links.	The Backup POSIX hard links as files option is not available on OpenVMS.
25	XFC Cache thrashing	ABS takes care not to thrash the XFC cache on OpenVMS	DP does not take care of avoiding the XFC cache and thus thrashes the XFC cache when a backup is performed.

Reporting

If the backup or restore occurs in a lights-off mode, for any backup solution, you can generate a report. These reports help you or the administrators to verify the status of backups. Both ABS and DP generates report and sends the notification.

In ABS, event notifications can be set for

1. Backup/restore status
2. Media (if number of usable media < threshold defined by user)

DP provides a highly customizable and flexible tool which provides a rich set of built-in reports. A user can choose from pre-configured reports and specify desired parameters for these reports. You can also select report formats (for example, ASCII, HTML). Some of the reports are as follows:

- Inventory/Status Reports
- Capacity Utilization Reports
- Problem Reports
- Event Based Reports

For more information

- ABS Document Set
- DP Document Set

References

- ABS Document Set
- DP Document Set
- Article on "abs-dataprotector_differences" by Ted Saul

MIKADO, a Virtual VAX Based on a Real-time Operating System

Robert Boers, CEO and CTO Stromasys SA, Switzerland
(For bio see http://www.stromasys.ch/about-us/board-of-directors)

Introduction

The Virtual Address Extension (VAX) processor family by Digital Equipment Corporation has earned its place in the history of computing. Two decades of development and sales exposed computer users to these well-designed and reliable systems, which were copied in former Eastern Europe as the best architecture available. However, with emerging technologies and the quest for ever faster, smaller and cheaper components, VAX hardware has become obsolete.

While computer hardware will eventually wear out, fall apart, burn up or simply stand in the way, its software can remain to be of significant value to its users. Vendors are of course delighted to sell new systems and might even provide software conversion tools. However, application migration is not trivial and is usually impossible if no source code is available.

Modern computer systems can execute complex tasks much faster than their ancestors of two decades ago. That makes it feasible to design a very precise software model of an older computer system and have it execute the original hardware instructions in software at the same or higher speed than the original. For nearly 15 years Stromasys SA (and its former incarnation as 'Software Resources International') has been building such 'virtual VAX' systems to replace aging hardware under the product name CHARON-VAX. (This technology is called 'processor emulation' in the terminology of the Gartner group).

CHARON-VAX products mostly use Windows as host operating system, as their real-time response is sufficient for most usage. Since the virtual VAX implementation provides a direct connection between the original VMS drivers and the peripheral hardware, we found that we could disable any interfering Windows services that would impact the reliability of the OpenVMS operating system, leaving windows to provide boot and file services.

The phenomenal development of computer hardware over the last decade made virtual legacy systems practical: Our first VAX emulator ran on a 300 MHz Celeron, just keeping up with a MicroVAX 3600. Today, on servers with a 3+ GHz CPU clock, the performance of a virtual VAX is 4 to over 100 times higher than the original hardware, depending on the virtual VAX model. On the I/O side, the higher system clock and the faster I/O subsystems of modern servers have reduced I/O latency and increased VMS disk I/O. A virtual MicroVAX disk controller can see file I/O increase by an order of magnitude.

There are situations however, where replicating precise application timing is more important than throughput when substituting VAX hardware with virtual VAX systems. Typical cases are industrial control systems and military applications. The presence of time critical I/O prevented replacement of such systems. We developed several techniques to adjust timing, so that also these systems can be candidates for replacement with a virtual system if a complete redesign and testing is to be avoided. The MIKADO project was one of these investigations.

MIKADO

In 2004 we designed the MIKADO system, a software model of a MicroVAX system that executes a MicroVAX hardware configuration in a QNX Neutrino real-time operating system (RTOS) running on I86 hardware[1]). MIKADO was a port of an existing CHARON-VAX Linux code base, and we retained the existing control and configuration components for convenience. The real-time

[1] The name 'MIKADO' was derived from the pick-up stick game. The design goal was that the manipulation of one component of Mikado would not influence the timing of the other parts.

environment promised a much lower inter-process latency compared to the then current Windows based systems.

In the early Windows based systems we used configurable timing parameters to adjust I/O response. With dedicated hardware (for example, Qbus adapters) and/or careful tuning we could replace VAX systems with special I/O requirements, at the cost of lengthy testing. The purpose of the MIKADO project was to understand the advantages of using an RTOS compared to Linux or Windows in building virtual hardware products. The prototype resulted – after a year of testing and modifications in a product that is still sold under the name 'FutureVAX'. FutureVAX is typically used in situations where a mixture of simulated and physical legacy peripherals require a specific, virtual system behavior that can be calibrated automatically.

It is interesting to see that at the time of writing this paper (2010) both emulator technology and host system performance have improved substantially, so that Windows or Linux based virtual VAX and Alpha products normally work 'out of the box'. In general, the interest in 'virtualizing' the legacy system is as much about preserving the OpenVMS environment on a standard server environment, as about removing the dependency on old hardware and avoiding application migration.

However, it does not mean that an RTOS based virtual legacy system has no role anymore. There are situations where an accurate replication of the original system performance and timing is required. Experience has shown that process control or military legacy systems were often 'tuned' to fix problems quickly, at the expense of proper code design. Operating system patches, adding NO-OPs to make custom drivers work or padding serial output with blanks to avoid implementing a handshaking protocol are difficult to fix if only the binary application code remains. While you can argue about the correctness of such implementations, as with all other legacy hardware, such systems will eventually have to be replaced.

The MIKADO advanced development project investigated several ways to calibrate application timing on virtual systems in a way that was independent of the performance of the host system. We had prior experience in system/application calibration with a virtual PDP-11 system where the user could specify the execution time of each individual instruction in any of its major modes. It worked but was complex to calibrate; the impact of individual instruction timing changes on a specific application is usually unpredictable.

In our experiments we found that, given enough system headroom, the actual abstraction of peripherals in a virtual system is not a problem. As long as the emulated I/O device can respond to the demands of the physical connection (serial bit rate, SCSI clock rate, and so on) in time, its absolute time to execute a command is usually not important. In a well designed virtual system nearly all the peripheral abstraction layers (with very few exotic exceptions) cause a minimal host system load.

The element determining the overall virtual system performance is normally the virtual CPU. In the early Windows based systems, the required context switching to serve virtual peripherals contributes to the system overhead and timing inaccuracies. At that time, MIKADO provided a much more predictable solution for I/O timing. However, the emergence of hyper threading, and later multicore systems largely removed that problem, at least for virtual systems with a single CPU.

The next step was to define a way to influence and measure I/O latency. For VAX CPU performance there is the traditional VAX 'VUPs' calculation. There is no established way to compare I/O latency between hardware and virtual systems. One of the experiments concerned an 'oscillator', using a parallel I/O card (DRV11J) with one output connected to an input. The output change triggered an input interrupt that changed the output again, using a standard VMS parallel

I/O driver and a (very short) applications program. We compared a hardware MicroVAX 3600 with an virtual VAX, using in the latter case a Qbus adapter (BCI 2100 from The Logical Company) to connect the DRV11J. We did similar tests via (emulated) printer ports.

Based on several experiments we concluded that the simplest way to adjust system performance is to suspend the virtual CPU for a variable time. The exact point where and how to suspend CPU performance required a great deal of thought and experiments; a good implementation (for example, handle pending interrupts first) determines the ultimate virtual system quality.

The CHARON-VAX code is written in C++, and did not pose a problem porting to QNX. The internal messaging structure was partly changed to use QNX specific features. Most of the code modifications took place in the virtual VAX CPU. The product licensing mechanism (a physical key) required a complete re-implementation as the product we used was not supported in an RTOS, but that part did not influence the virtual VAX functionality. The work resulted in a well tunable virtual VAX system on QNX, but we were still missing the tools to easily tune the performance and automate the process, if desired.

Incidentally, where in our virtual PDP-11 the extra microsecond delays in instruction execution were too short for a context switch, a suspension of a few milliseconds in the right moment allows the host CPU to attend to other tasks, or offload the host system. Without specific measures, the virtual VAX CPU will run the VMS idle loop, loading the host CPU to 100% whenever it gets CPU cycles available. The work on calibration software provided a means to detect the VMS idle loop and suspend virtual CPU operation. We put these findings back into our Windows and Linux based products. This provides the feature that running a lightly loaded VMS environment using a virtual VAX or Alpha on a laptop will not drain the laptop battery at maximum speed.

As a result of the MIKADO project, we developed the FutureVAX product (a QNX based virtual VAX), which supported replacement Qbus hardware and an embedded PC on a quad Qbus card form factor, designed by The Logical Company. The latter is essentially a plug-and-play solution to replace most of the components of a MicroVAX, but retain the existing enclosure and any special I/O hardware.

Tools for calibrating virtual systems

The final phase of the MIKADO project was to build an easily usable calibration tool, including the possibility for the virtual system to calibrate itself. That means that the virtual system can measure its own performance during run-time and can change its emulated system performance accordingly. In this way, a virtual system and its applications could change to a different host platform (including a VMware client) reset itself to the desired performance. Note that automatic calibration is currently rarely used, but we plan to use it in future products for the industrial automation market.

Automatic calibration raises an interesting dilemma. A virtual VAX is designed as an exact replica of the original hardware. The CHARON-VAX management console, which controls proper operation of the emulated environment, sits on the 'other side' of the virtual layer, and is not visible from VMS or its applications. But it is the management console that has to change some component parameters to change the virtual VAX performance dynamically. This requires a communication channel across the virtual system layer.

There are several ways to create such a communication channel. One way is to create a special peripheral (for instance a serial port) that does not connect to the outside world but into the CHARON management console. However, such a solution could interfere with an existing application and is guest system specific (we want to use the same mechanism for other virtual

platforms). The solution we chose was adding extra architectural registers to the virtual CPU, which are obviously not used by any software coming from a physical VAX. A small OpenVMS software utility (see appendix) provides the connection to the CHARON-VAX management console.

The selected unit of delay in the virtual VAX CPU is long enough to allow a context switch in Windows. In QNX the context switch is much faster, but we wanted the test package have comparable characteristics on all platforms. As the appendix shows, by increasing the number of delay units we obtain a virtual system performance curve. We found that the 'VUPs curve' and the frequency of our 'oscillator test' followed approximately the same curve, leading to the assumption, that carefully slowing down the virtual CPU is a workable method to adjust critical I/O timing. Testing with some known difficult applications confirmed this.

The slowdown facility is integrated in many of the CHARON-VAX products. The MIKADO package allows the slow down factor to be set in static mode (in the CHARON-VAX configuration file) or dynamically (by calling SLOWDOWN or SPEEDUP from an OpenVMS application). And our current products also inherited the power save mode when the guest operating system on our virtual platform is idle (for those operating systems where we can detect an idle mode) and a facility for the emulated VAX to shut down its host system if all the work is done. These features reduce system load on (blade) servers as well.

Thanks to their unsurpassed reliability, there are still many VAX VMS systems in the world that control power plants and distribution networks, railroads, steel and automobile plants, military test systems, and other applications with strict real time requirements. Unlike administrative systems, their connection to other equipment made replacement with virtual VAX platforms more complex. However, after more than 30 years hardware parts become scarce, and often the knowledge about these critical systems is no longer present in a company. We now see an increasing demand for replacing such VAX and even PDP11 systems, probably the tail end of the installed base. Fortunately, the power of modern servers and a better understanding of how to virtualize such systems provide the means to preserve these software investments across hardware generations.

Appendix: The MIKADO CPU calibration package

The MIKADO CPU calibration package is a collection of utilities to influence the execution speed of the virtual VAX in a precise manner. While in general the fastest possible code execution is desirable, the implementation of real-time systems can require a specific VAX performance. Host hardware varies widely in execution speed, and setting performance will usually require the combination of an emulator calibration phase followed by a programmed slowdown.

When the host system resource use must be managed, it is useful to reduce the emulator load on the host system when the guest OS (usually VAX/VMS) is idle. It can also be desirable to terminate virtual VAX instance after the guest OS terminates, for instance to release the MIKADO disk images for automated backup. To permit calibration for individual VAX/VMS applications and instances, the performance control can be handled from within the VMS environment by means of VAX/VMS utilities which directly interact through the emulated VAX system with the MIKADO management system.

The MIKADO calibration package provides the following performance calibration functions, running as OpenVMS applications on the virtual VAX:

CALCULATE_VUPS.COM VUPS meter, used with the SLOW_TEST system calibration mechanism.

SLOWDOWN.EXE Slow the emulator one step.

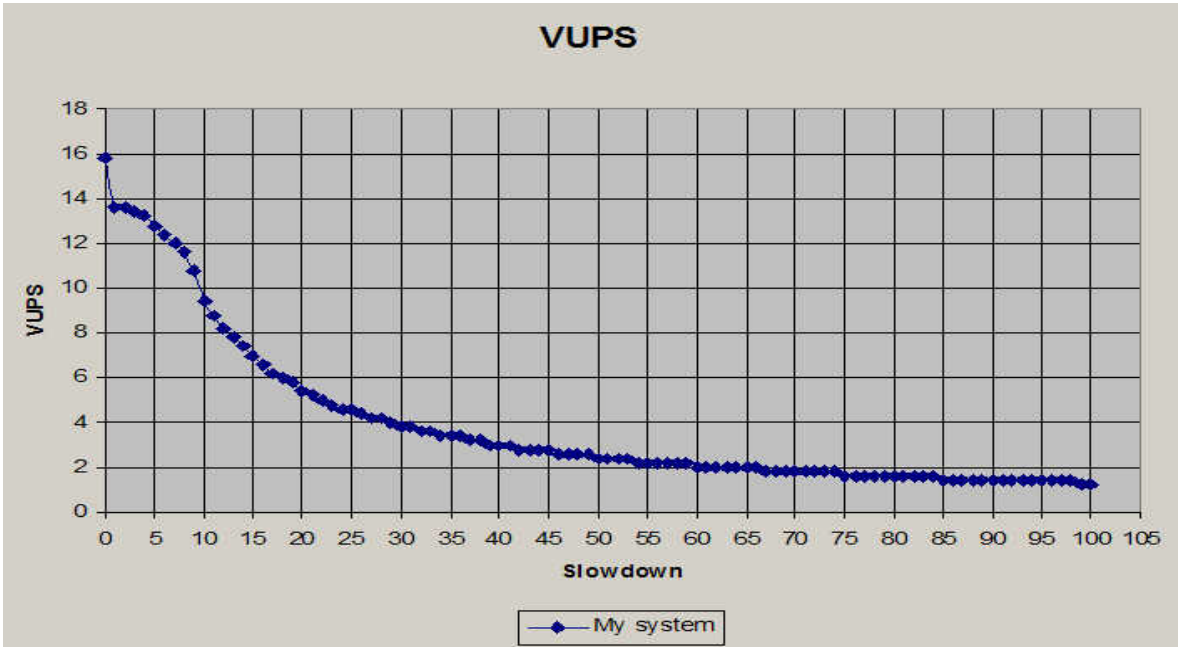
SLOWDOWN_R.EXE	Remove the execution slowdown – run in full speed.
SLOW_TEST.COM	Calibrates the slowdown mechanism.
SPEEDUP.EXE	Increase the emulator speed one step.

The actual relative system performance can be measured by executing SLOW_TEST.EXE at the VMS prompt (SLOW_TEST.EXE uses the CALCULATE_VUPS.EXE utility). The generated slow_test.log file expresses the VUPs performance in terms of slowdown 'steps', which are reductions of the virtual CPU clock, where the value 0 corresponds to unmodified performance:

```
Calculating at slowdown 1...
   Approximate System VUPs Rating :  13.4
Calculating at slowdown 2...
   Approximate System VUPs Rating :  13.2
```

…

The result of the slowdown test lists the relationship between the slowdown value and VUPS value on the virtual VAX, as illustrated in the following example:



The desired slowdown factor can be specified in the MIKADO configuration file. Using the above example, if you want the system to run at a performance of ~ 4 VUPS, you can just add to the configuration file: set idle slowdown=29

The relationship between the slowdown rate and emulator performance is non-linear, and the VUP measurement fluctuations can create apparent small increases in performance when the slowdown mechanism is increased one step. On virtual systems, the VUPs measurement is an approximation with a statistical variation (maximum 5-10%) increasing with a larger emulated memory size. It will be rarely necessary to remove the statistical error, but if needed you can run the calibration multiple times and calculate the standard error at each point.

Assuming that the maximum error is approximately 10% of the value, a 4 VUPS setting will provide 3.6 - 4.4 VUPs. Note that the VUPs value does not indicate I/O performance; it is only a measure

of the VAX CPU - memory interaction. The precision of the performance slowdown increases with an increase of the slowdown parameter. This suggests that using a fast host system provides a more precise environment.

Dynamic performance calibration

The MIKADO VAX emulator performance depends on various factors: the host system CPU(s), its memory architecture, use of the host system CPU for other applications, the actual MIKADO configuration, the peripheral interrupts it handles and the VAX operating system (typically VMS) configuration. Consequently, while the approximate emulator performance can be predicted based on the host platform, calibration will be required for a more accurate setting.

With an insertion of a slowdown parameter in the MIKADO configuration file, as described above, this setting will apply to the whole configuration during its total runtime. However, MIKADO provides a real-time connection between the OpenVMS operating system and the emulator component manager. For dynamic performance adjustment call SLOWDOWN.EXE and SPEEDUP.EXE at the VAX/VMS level to set the right performance in any critical step of a workload execution. SLOWDOWN_R.EXE can be dynamically used to remove any slowdown. By including CALCULATE_VUPS.COM (or application specific responses) in the performance management, the guest VAX/VMS OS can auto-calibrate its execution speed at any stage of application execution.

Host system control

The MIKADO calibration package provides the following host system control functions, running as OpenVMS applications on the virtual VAX*:*

IDLE.EXE	Reduces host system load when the Guest OS is idle.
SHUTDOWN.EXE	Shuts down the emulator in 30 seconds.
SHUTDOWN3.EXE	Shuts down the emulator in 3 minutes.
SHUTDOWN5.EXE	Shuts down the emulator in 5 minutes.
SHUTDOWN_R.EXE	Remove scheduled emulator shutdown.

The IDLE utility is required for all calibration functions, as it provides the connection to the virtual VAX management console.

For more information

www.stromasys.com

OpenVMS Technical Journal V16

OpenVMS RAD Support on Integrity Servers

Shyamalendu Sarkar
Durga Prasad P V
Narayanan A N

Introduction

OpenVMS V8.4 introduced support for Resource Affinity Domain (RAD) for Integrity servers with Non-Uniform Memory Architecture (NUMA). Cell-based Integrity servers (rx7620, rx7640, rx8620, rx8640 and Superdomes) and Integrity i2 servers (BL860c i2, BL870c i2, BL890c i2, rx2800 i2) are all based on the NUMA architecture.

In OpenVMS, a software grouping of hardware processing resources and memory with similar access characteristics is called a RAD.

The RAD support functionality is same on the Alpha NUMA servers running OpenVMS V8.4.

This article discusses the key aspects of RAD support in OpenVMS V8.4 and its subsequent software updates for Integrity servers.

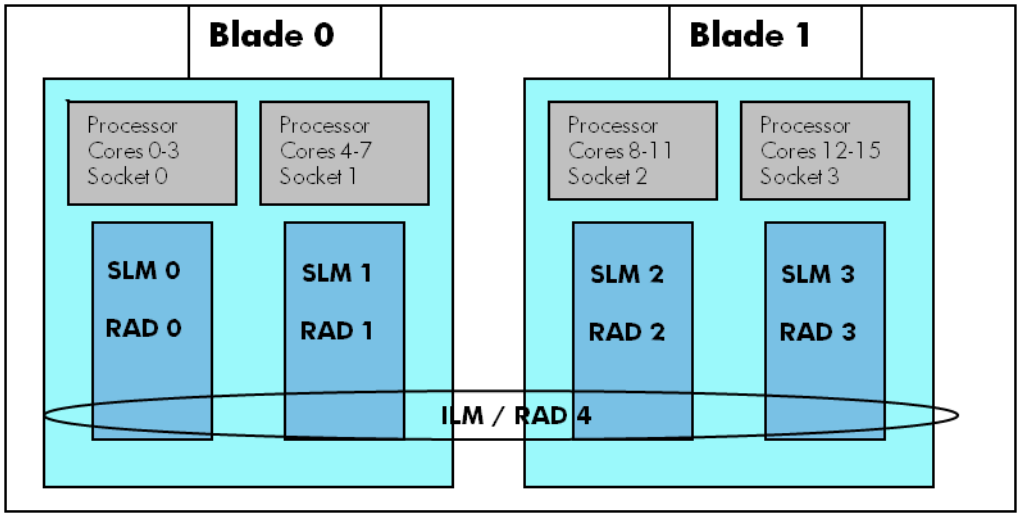
Background

Socket Local Memory

Integrity i2 servers are based on the quad-core or dual-core Tukwila processors. Each CPU socket is coupled with specific memory Dual Inline Memory Modules (DIMMs) through its integrated memory controllers. This memory is termed as Socket Local Memory (SLM).

Access to memory local to a socket is faster when compared to access to memory in a remote socket (other socket in the same blade or another blade). Figure 1 illustrates an i2 server (BL870c i2) configuration having two blades (blade 0 and blade 1). Each blade contains two quad-core sockets with their socket local memory. Memory access for socket 0 from SLM0 is faster when compared to access from SLM1, even though both the sockets are in the same blade. Similarly, memory access latency across the blades is costlier.

Figure 1: BL870c i2 Configuration with SLM/ILM and RAD

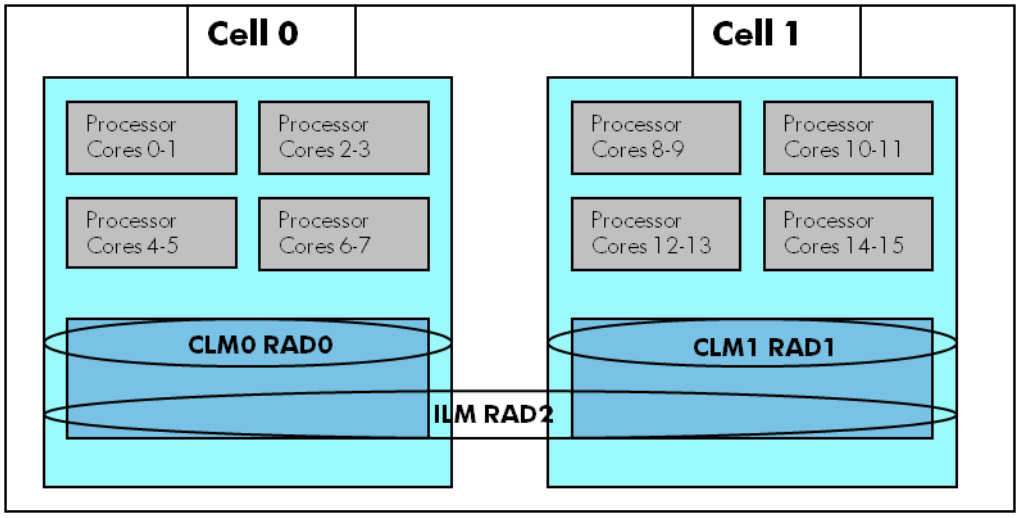


Cell Local Memory

On cell-based servers, processors and memory are grouped together into Cells. This memory is known as Cell Local memory (CLM). Access to memory local to a cell is faster when compared to access to memory outside a cell (remote cell).

Figure 2 illustrates an example for configuring two cell rx7640 system with CLM and ILM.

Figure 2: RAD Configuration on two cell rx7640 with CLM and ILM



Interleaved Memory

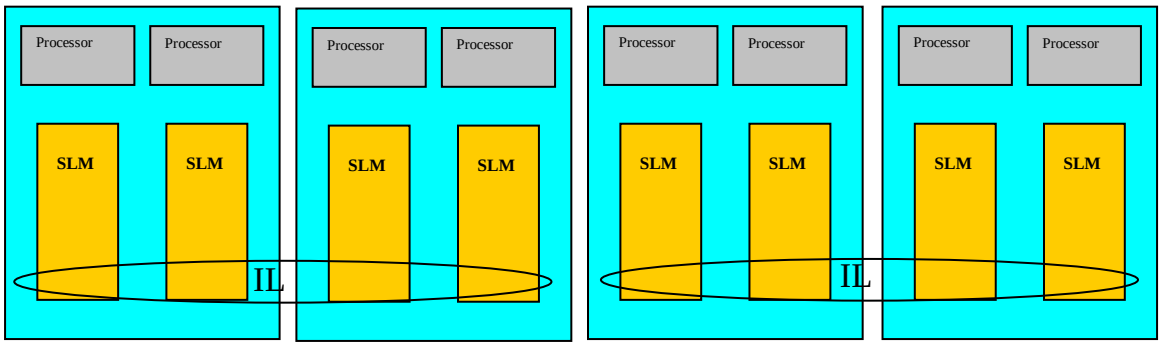
A portion of memory is taken from all the sockets or cells in a server, and it is striped in a round-robin fashion of cache-line sized chunks. It has the characteristic that memory accesses are on an average uniform. This memory is termed as the Interleaved Memory (ILM).

ILM provides consistency to memory access, irrespective of which processor in the system is accessing it.

With the exception of BL890c i2, all the other NUMA Integrity servers can have a single ILM. BL890c i2 can have two ILMs, with each ILM spanning across a set of two blades – ILM 0 across blade 0 and blade 1, ILM 1 across blade 2 and blade 3.

Figure 3 illustrates an example for configuring BL890c i2 system with SLM and ILM.

Figure 3: BL890c i2 system with SLM and ILM



How to configure memory for NUMA?

Memory configuration with only SLM/CLM or a combination of SLM/CLM and ILM is possible on Integrity NUMA servers. However on a cell-based system, OpenVMS requires a minimum of 2 GB of ILM.

A change in the memory interleaving configuration on i2 servers is accomplished using the `memconfig` command. The system requires a reset for the change to take effect.

```
Shell>memconfig –mi
```

OpenVMS supports the following memory interleaving configurations:

- MaxUMA (0% SLM:100% ILM)
- MostlyUMA (12.5% SLM:87.5% ILM)
- Balanced (50% SLM: 50% ILM)
- MostlyNUMA (87.5% SLM:12.5% ILM)
- MaxNUMA (100% SLM: 0% ILM)

Note:
MostlyNUMA is the default memory configuration.

Integrity cell-based systems need to be configured to have CLM; otherwise the entire memory is treated as interleaved (ILM).

HP recommends using Partition Manager (`parmgr`) for configuring memory on the Integrity cell-based systems. The `parmgr` command provides a graphical user interface and is available for free from HP. For more information on the `parmgr` command, see the Technical documentation website at:
http://bizsupport2.austin.hp.com/bc/docs/support/SupportManual/c02058520/c02058520.pdf

Resource Affinity Domains

In OpenVMS, a software grouping of hardware processing resources and a memory with similar access characteristics is called RAD. Each RAD is a logical grouping of processors and memory local to a socket or a cell. OpenVMS treats the ILM as a separate RAD; also all the processors in the system are considered as members of this ILM RAD.

Base RAD

The ILM is considered as the BASE RAD. The BASE RAD contains all the system data that is not specific to a process or to a CPU, including read-only code and read/write data. If the ILM is not configured in the system, RAD 0 is treated as the BASE RAD.

A BL890c i2 server blade can have two ILMs, each of which corresponds to a RAD. The first ILM RAD is designated as the Base RAD.

Home RAD

Each process, when created, is assigned to a particular RAD and this RAD is considered as the HOME RAD for that process. OpenVMS automatically assigns a Home RAD for every process if a specific RAD is not explicitly mentioned for that process. By default, OpenVMS assigns one of the RADs other than the Base RAD as a process Home RAD.

OpenVMS Memory Manager and RAD

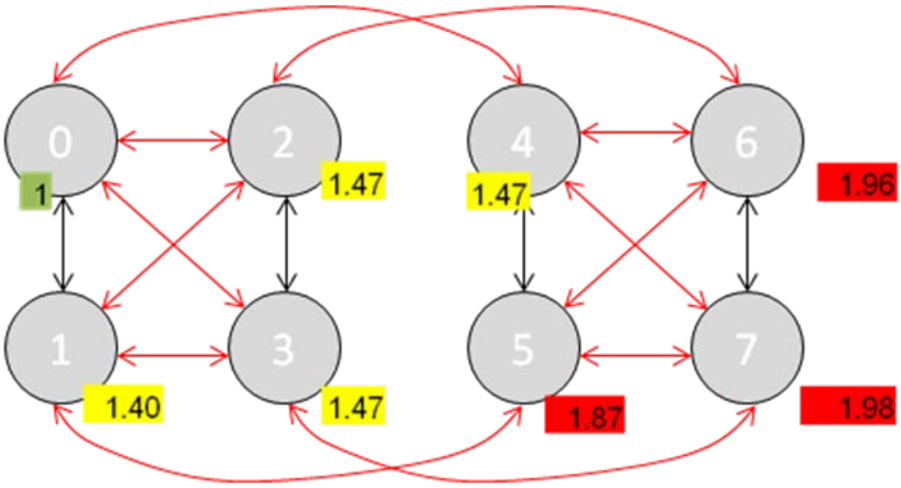
The OpenVMS Memory Manager allocates the required memory for a process from the Home RAD of that process. If this process runs on a CPU from its Home RAD, the number of local memory references will far exceed the number of remote memory references. This helps to reduce the CPU wait cycles for memory references, thereby increasing the system performance.

RAD Preference Array

On Integrity servers, OpenVMS maintains a RAD Preference Array for each RAD. Entries in this array are ordered in terms of increasing access cost. During memory allocation, this array is used to choose the optimum RAD(s) to satisfy the memory requirement. If the memory is not available in a particular RAD, the RAD Preference Array is used to determine the next best RAD with minimum access cost from where memory can be allocated.

Figure 4 illustrates the relative memory latencies for a CPU in the Socket 0 across various SLMs in a BL890c i2 server blade – access to SLM 0 being the fastest and access to SLM 7 being the slowest.

Figure 4: Relative memory latencies across Sockets on BL890c i2



- Slowest latencies are marked in red
- Second slowest latencies are marked in yellow
- Fastest latencies are marked in green

The following table lists the sample RAD Preference Array for each RAD on a BL890c i2 system with SLMs and ILMs.

RAD Preference Array for a BL890c i2 system with SLMs and ILMs

RAD #	RAD Preference Array
0 (SLM 0)	0; 8(1st ILM);1;2;3;4;5;9(2nd ILM);6;7
1 (SLM 1)	1; 8(1st ILM);0;2;3;5;4;9(2nd ILM);6;7
2 (SLM 2)	2; 8(1st ILM);3;0;1;6;7;9(2nd ILM);4;5
3 (SLM 3)	3; 8(1st ILM);2;0;1;7;6;9(2nd ILM);4;5
4 (SLM 4)	4; 9(2nd ILM);5;0;6;7;1;8(1st ILM);2;3
5 (SLM 5)	5; 9(2nd ILM);4;1;6;7;0;8(1st ILM);2;3
6 (SLM 6)	6; 9(2nd ILM);7;2;4;5;3;8(1st ILM);0;1
7 (SLM 7)	7; 9(2nd ILM);6;3;4;5;2;8(1st ILM);0;1
8 (ILM 0)	8(1st ILM);0;1;2;3;9(2nd ILM);4;5;6;7
9 (ILM 1)	9(2ndILM);4;5;6;7; 8(1st ILM);0;1;2;3

Page Zeroing

In a RAD-enabled system, page lists corresponding to free pages and demand-zero pages are maintained individually for each RAD. The physical pages in these lists correspond to the SLM/CLM and ILM of the associated RADs.

CPU, when in its idle loop, zeroes free pages of physical memory. This meets the need for future demand-zero page allocations. For Page Zeroing, the OpenVMS Memory Manager chooses pages from the Home RAD of the idle CPU till sufficient number of pages are zeroed. This ensures local memory references, thereby keeping the latency to a minimum.

Additionally, once a sufficient number of demand-zero pages are available in a RAD, a CPU executing in the idle loop in that RAD assists in zeroing the pages for the ILM RAD. In BL890c i2 server blade that has two ILM RADs, the CPU does the Page Zeroing for the ILM RAD closer to it.

Per-RAD Non-Paged Pool

On a RAD-enabled system, the Non-Paged Pool can be configured for each RAD. By default, this feature is turned off in the Integrity NUMA servers and the Non-Paged Pool comes from the Base RAD. In the Alpha servers with RAD configured, this feature is turned on by default.

If the sixth bit in the RAD_SUPPORT system parameter is set, the Per-RAD Non-Paged Pool feature is enabled and the NPAGERAD system parameter value is used to determine the total memory for the Non-Paged Pool among all the RADs other than the Base RAD. If the value of the NPAGERAD is 0 (default), OpenVMS computes a suitable value based on the distribution of memory between the SLMs/CLMs and the ILM.

For more information on NPAGERAD system parameter, see References.

The NPAGEDYN system parameter defines the total amount of Non-Paged Pool memory on the system. The Non-Paged Pool memory size from the Base RAD is (NPAGEDYN – NPAGERAD).

OpenVMS Scheduler and RAD

With RAD soft-affinity, the OpenVMS Scheduler attempts to schedule a process on a CPU from the Home RAD of that process. This is a complementary mechanism to ensure that when a process is scheduled on a CPU, the memory references for that process become more local. Any CPU hard-affinity set for a particular process to particular CPU(s) takes precedence over the RAD soft-affinity.

Optimized Scheduling Algorithm

When a process is ready to be scheduled, if the scheduler does not find an idle CPU from the Home RAD of that process, it skips the scheduling of that process for 'skip count' iterations; beyond that the Scheduler chooses a CPU from a remote RAD.

For more information on RAD_SUPPORT system parameter, see References.

RAD—DCL commands

OpenVMS provides various DCL commands and Lexicals to view and modify the Home RAD of a particular process in the system. The following examples illustrate the same:

Command to display the Home RAD of the current process

```
$ show proc/rad

18-APR-2010 02:17:31.49   User: SYSTEM      Process ID:   0000042F
Node: SKD12                                 Process name: "SYSTEM"

Home RAD: 1
```

Command to display the Home RAD of a particular process

```
$ show proc/rad/id=00000421
18-APR-2010 02:18:45.52   User: INTERnet    Process ID:   00000421
Node: SKD12                                 Process name:
"TCPIP$INETACP"
Home RAD: 1
```

Command to change a process' Home RAD

```
$SET PROCESS/RAD=HOME=n
```

Note:
A process' Home-RAD is changed implicitly when its affinity is set.

For example:

```
$ show process/rad

18-APR-2010 19:48:30.75   User: SYSTEM    Process ID:   0000042F
```

```
Node: SKD12                                   Process name: "SYSTEM"

Home RAD: 1
```

If this process' affinity is set to any processor belonging to RAD 0, its Home RAD is changed to RAD 0.

CPU 1 is from RAD 0.

```
$ set process/affinity/set=1

$ show process/rad

18-APR-2010 19:54:43.21   User: SYSTEM    Process ID:   0000042F
Node: SKD12                               Process name: "SYSTEM"

Home RAD: 0
```

Here the process Home RAD is changed from RAD 1 to RAD 0.

Note:
HP does not recommend frequent changes of RAD and CPU affinity as it can degrade the performance.

Utility to understand the distribution of memory and CPUs across RADs

```
$ @SYS$COMMON:[SYSHLP.EXAMPLES]RAD.COM;

Node: OVMS Version: V8.4      System: HP rx7640 (1.60GHz/12.0MB)

RAD   Memory (GB)   CPUs
===   ===========   ===============
  0       27.49     0-7, 16-23
  1       35.49     8-15, 24-31
  2        8.99     0-31
```

RAD—SDA commands

The SDA commands related to RAD are as follows:

- Command to get the details of a system RAD configuration.
  `SDA> show rad [number|/all]`
- Command to get the page distribution across RADs in terms of free pages and zero pages.
  `SDA>show pfn/rad`

Importance of RAD

The NUMA architecture provides mechanisms to reduce the average memory latency. The magnitude of the reduction of this memory latency for a process depends upon the memory access pattern of the process and the memory configuration in each RAD. Local memory access is much

faster when compared to a remote access or access to the ILM. The best possible RAD configuration – proper percentage of SLM/CLM and ILM distribution – is one of the key factors for performance improvement. Accessing the local memory reduces the processor cycles spent on memory references, thereby reducing the processor utilization and increasing the throughput.

The impact of RAD becomes more when we deal with a large system, for instance BL890c i2.

Figure 4 illustrates relative latencies on BL890c i2. From the Socket 0, accessing memory in the Socket 7 is much more expensive when compared to accessing memory in the Socket 1 (in the same blade).

When to use RAD?

Most of the enterprise-application software exhibits high amount of local memory reference. Few applications are designed to access more global data and have scattered access to memory. RAD is best suited for the first category of applications. For the latter, usage of Interleaved Memory is a better choice than Local Memory.

In some cases, the number of applications deployed on the system and the layout of the process distribution has a higher bearing on the RAD. For instance, if a system is hosting multiple applications with small working sets compared to the available physical memory, RAD is the obvious choice. With a small working set, application processes will have a better locality of memory reference and performance gain. On the other hand, if a single application with a very large working set is deployed on a system, impact of RAD will be less.

RAD configuration guidelines

Applications with more local memory reference pattern and small working set will gain good amount of performance, if most of the memory is configured as Socket Local Memory or Cell Local Memory.

On the BL8x0c i2 server, in our in-house tests with Oracle, the MostlyNUMA memory configuration exhibits the best performance.

On a cell-based system, HP recommends a composition of 7/8 of total memory as CLM and 1/8 of total memory as ILM. However, if the application exhibits a performance degradation, either the application is suitable for fully Interleaved Memory or the system requires further tuning in the RAD configuration.

RAD performance monitoring

On RAD configured systems, the `RADCHECK` utility of OpenVMS provides information on the locality of memory in terms of page distribution across RAD(s) for a process.

This utility can control the RAD scheduling algorithm and provide the scheduling statistics.

```
$ radcheck :== $sys$test:radcheck
```

To get more information on the usage of `RADCHECK`, use the help option.

```
$ radcheck  -help
```

References

- RAD Support in OpenVMS V8.4
  http://h71000.www7.hp.com/doc/84final/6679/6679pro_002.html#index_x_35

- HP OpenVMS System Management Utilities Reference Manual: (M–Z) –
  http://h71000.www7.hp.com/doc/84final/6048/ba555_90009.pdf

- Memory subsystem information for HP Integrity Server Blades (BL860c i2, BL870c i2, and BL890c i2)
  http://h20195.www2.hp.com/V2/GetPDF.aspx/4AA1-1126ENW.pdf

OpenVMS Technical Journal V16

OpenVMS Guest Troubleshooting

Bhadresh Udani

Introduction

HP Integrity Virtual Machines (Integrity VM) supports OpenVMS for Integrity servers Version 8.4 is supported as a guest operating system. Integrity VM is a soft partitioning and virtualization technology within the HP Virtual Server Environment, which enables you to create multiple virtual servers or machines with shared resources within a single HP Integrity server or a hardware partition (nPartition).

Each virtual machine hosts its own "*guest*" operating system instance, applications, and users. On HP Integrity servers, the Integrity VM Host runs on HP-UX, while OpenVMS runs as a *guest* operating system.

This article provides information on some of the common tools used for Troubleshooting or Debugging OpenVMS Guests. The article describes basic configurations, common failure scenarios, and tips on how to handle them.

Installation requirements

The following table lists the requirements for installing VM Host system for Integrity VM and OpenVMS guest on VM Host System.

Resource	Description
Server	Integrity Server VT-i (Intel Virtualization Technology for the Intel Itanium Architecture) enabled Intel Itanium processors
Host operating System	HP-UX 11i v3 March 2010 and above
Guest operating system	OpenVMS V8.4
LAN card	Required for network connection and configuration
Software	Minimum Requirement: Integrity VM Version 4.2 and the following patches. The patches are available at: http://itrc.hp.com. • PHSS_40875 1.0 HPVM B.04.20 CORE PATCH • PHSS_40876 1.0 HPVM B.04.20 VMAGENT • PHSS_40901 1.0 HPVM B.04.20 VMMIGRATE PATCH

Integrity VM installation

This section describes the installation and verification process for Integrity VM.

For HP-UX 11iv3 March 2010 installation, see the *HP-UX 11i v3 Installation and Update Guide* at:

http://bizsupport2.austin.hp.com/bc/docs/support/SupportManual/c02023874/c02023874.pdf

Check for hyperthreads

Integrity VM Version 4.2 does not support hyperthreading. To disable hyperthreading, use the following command:

```
# /usr/sbin/setboot -m off
# reboot
```

Or

```
SHELL> cpuconfig threads off
SHELL> reset
```

Note:
Be sure that the hyperthreads are disabled. Use the following command to check if the hyperthreads are enabled:

```
# getconf SC_HT_ENABLED

1 indicates hyperthreading is enabled
0 indicates hyperthreading is disabled
```

Or

```
SHELL> cpuconfig threads
```

Installing Integrity VM

To install Integrity VM, enter the `swinstall` command (assuming the kit is at /var/depots).

```
# swinstall -x autoreboot=true -s /var/depots/hpvm4.2.depot T2767CC
```

Verifying Integrity VM installation

To verify that Integrity VM is installed successfully, enter the following command:

```
# hpvminfo
hpvminfo: Running on an HPVM host.
```

Note:
To see which version of specific bundles is installed, enter the `swlist` command. Specific version numbers on your installation might not match with the examples described in this document.

```
# swlist | grep Integrity

T2767CC            B.04.20       Integrity VM
T8718AC            B.04.20       Integrity VM Online Migration Software
VMGuestLib         B.04.20       Integrity VM Guest Support Libraries
VMGuestSW          B.04.20       Integrity VM Guest Support Software
VMKernelSW         B.04.20       Integrity VM Kernel Software
VMProvider         B.04.20       WBEM Provider for Integrity VM
```

To manually start and stop Integrity VM, use the following commands:

```
# /sbin/init.d/hpvm start
# /sbin/init.d/hpvm stop
```

OpenVMS guest troubleshooting

This section describes about the failure scenarios while installing or configuring an OpenVMS guest, and provides troubleshooting tips.

Integrity VM is not running

```
# hpvmstart –P OVMSG1
hpvmstart: HPVM currently not running.
hpvmstart: Unable to continue.
```

The error indicates Integrity VM is not running. To start Integrity VM, use the following command:

```
# /sbin/init.d/hpvm start
```

VIO devices are not supported

```
# hpvmmodify -P OVMSG1 -a disk:scsi::disk:/dev/rdisk/disk4
```

HPVM guest OVMSG1 configuration problems:

Warning 1 on item /dev/rdisk/disk4: Devices on OpenVMS guests must use the avio_stor adapter.

These problems might prevent the HPVM guest OVMSG1 from starting.
hpvmmodify: The modification process is continuing.

```
# hpvmmodify -P OVMSG1 -a network:lan::vswitch:myswitch
```

HPVM guest OVMSG1 configuration problems:

Warning 1 on item myswitch: Devices on OpenVMS guests must use the avio_lan adapter.

These problems might prevent the HPVM guest OVMSG1 from starting.
hpvmmodify: The modification process is continuing.

These warnings indicate that OpenVMS guest supports only the Accelerated Virtual I/O (AVIO) interface. Integrity VM commands enables you to configure the VIO devices to a guest and these devices might not give any errors during the startup. However, the VIO devices are not part of the supported configuration of a *guest* running on an OpenVMS operating system.

Delete the virtual switch (vswitch) and the disk with the SCSI/LAN Adapter and add the vswitch and disk with an AVIO adapter to resolve the issue as mentioned in the following section:

```
# hpvmmodify -P OVMSG1 -d disk:scsi::disk:/dev/rdisk/disk4
# hpvmmodify -P OVMSG1 -d network:lan::vswitch:myswitch
# hpvmmodify -P OVMSG1 -a disk:avio_stor::disk:/dev/rdisk/disk4
# hpvmmodify -P OVMSG1 -a network:avio_lan::vswitch:myswitch
```

Error starting Integrity VM

Cannot open VM
/var/opt/hpvm/uuids/6b3e85bc-334b-11df-8701-00306ef445ed/vm_dev: (12) Not enough space hpvmstart: ERROR (OVMSG1): Unable to start guest 'OVMSG1'.
hpvmstart: Unable to start guest 'OVMSG1'.
hpvmstart: Unable to continue.

This error occurs on any 'basic' error in getting the guest software (monitor) to load or Host hardware does not support to run OpenVMS guest.

You can use the –s option with the `hpvmstart` command to find the configuration issue. The -s option examines the specified guest configuration and reports any errors or warnings that are preventing it from starting. The guest is not started.

For example:
```
# hpvmstart -s -P OVMSG1
```
HPVM guest OVMSG1 configuration problems:
    Warning 1: OpenVMS guests not supported on this host's CPU type (Madison).
These problems might prevent the HPVM guest OVMSG1 from starting.
hpvmstart: Unable to continue.

Virtual switch does not start

Add virtual switch for guest OVMSG1:
```
# hpvmmodify –P OVMSG1 –a network:avio_lan::vswitch:Myswitch
```

The problem is with a virtual switch. The following section provides some information about the virtual switches:
```
# hpvmnet

Name     Number State     Mode     PPA      MACAddress    IP Address
======== ====== ======= ========= ====== ============== ===============
localnet    1    Up        Shared N/A             N/A
myswitch    2    Up        Shared lan1   0x001a4b087d231  11.11.111.111
```

It looks like the vswitch named Myswitch does not exist (though myswitch does). Virtual switch names are case sensitive, so it is often a common mistake to switch upper and lowercase letters in a vswitch name.

Virtual switch does not respond

- Check the log files:
  - Check guest's log and command.log for anomalies
- Check the performance of the virtual switch:
  - Examine output of the `hpvmnet –S <vswitch-name> -V`
  - Run it twice, counters are reset between execution

For example:
a. Looking at myswitch information:

```
# hpvmnet -V -S myswitch
Name : myswitch
.
.
.
Packets out to stream   : 5991
Packets out to guest    : 2455
Packets dropped         : 54
Broadcasts              : 4310

.
.
.
```

b. Run hpvmnet again to reset the counters and see if any packets are going out to stream:

```
# hpvmnet -V -S myswitch
Name : myswitch
.
.
.

Packets out to stream   : 0
Packets out to guest    : 235
Packets dropped         : 0
Broadcasts              : 235

.
.
.
```

c. The packets in and packets out to guest values should be greater than zero, this indicates the current activity on the vswitch. If it is greater then zero, then the vswitch is operational.

You can notice that nothing is going out to stream on that interface. Check the physical interface on the host. Run the `hpvmnet` command to find about the physical point of attachment (PPA).

```
# hpvmnet

Name     Number State  Mode   Name PPA  MAC Address    IPv4 Address
======   =========== ======= ========   ============   =============
localnet  1     Up     Shared    N/A         N/A
myswitch  2     Up     Shared   lan1     0x001a4b087d23  11.11.111.111
```

The output indicates that the PPA is lan1.

```
# ifconfig lan1
lan1: flags=843<BROADCAST,RUNNING,MULTICAST>
inet 11.11.111.111 netmask fffff800 broadcast 11.11.111.255
```

This indicates the PPA is down. If the state is UP, the output will be:

```
# ifconfig lan1
lan1: flags=843<UP,BROADCAST,RUNNING,MULTICAST>
inet 11.11.111.111 netmask fffff800 broadcast 11.11.111.255
```

d. First restart the vswitch, because any changes to the PPA are reflected in the vswitch. When you restart the vswitch, the PPA state is automatically set to UP.

```
# hpvmnet -r -S myswitch
hpvmnet: Restart the vswitch 'myswitch'? [n]: y
```

Guest cannot join in the cluster

You might see the following message while loading licenses in guest node, if it is a cluster member:

```
%LICENSE-I-LICIGN, 'product' 'authorization' License Ignored Not
Virtual
```

There is a new /VIRTUAL qualifier to load Per Core License (PCL) licenses in a guest environment when the guest node is a cluster member. Modify the licenses using the /VIRTUAL qualifier and reload it. The guest node will be able join as a cluster member after reloading the licenses.

```
$ LICENSE MODIFY 'product' /VIRTUAL
$ LICENSE LOAD 'product'
```

Guest crashes or hangs

If a *guest* operating system hangs, do one of the following to generate the dump:

- Use Ctrl/P to generate the OpenVMS crash dump.
  Or
- Use Ctrl/B to enter the virtual console. Set tunables and enter the `tc` command to reset the guest at CM.

  ```
  vMP:CM> say 0xb reco
  vMP:CM> say 256 rnum
  VMP:CM> say 1 dump
  VMP:CM> say 1 dump
  vMP:CM> tc
  ```

  The `tc` command generates an OpenVMS crash dump and the vm.core file. The vm.core file can be found at /var/opt/hpvm/guests/<guest>.

  Collect the INIT log after restart:

  ```
  vMP:CM> ed –init
  ```

  Alternatively, you can use the `hpvmconsole` command to pass commands to a guest console from the Host command line. For example, to force an INIT on a guest, use the following command:

  ```
  # hpvmconsole –P OVMSG1 –q –c tc –fi
  ```

  From the Host command line, you can save a copy of the INIT log:

  ```
  # hpvmconsole –P OVMSG1 –q –c 'ed –init' > /tmp/tc.log
  ```

If a Host hangs or guest crashes, output of the following form is displayed on the guest's virtual console:

```
*** A fatal error has occurred -- VM terminated ***
**** Dumping Guest Image ****
**** Done with dump (nnnnnKbytes) ****
```

- See the HPVM monitor log, located on the VM Host in the /var/opt/hpvm/common/hpvm_mon_log.
- After the guest reboots, use the `hpvmcollect` command to collect the information and log files. Report the information through your support channel.
- To collect logs, use the following commands:
  ```
  # hpvmconsole -P OVMSG1 -q -c cl > cons.log
  # hpvmconsole -P OVMSG1 -q -c 'ed -init' > init.log
  # hpvmconsole -P OVMSG1 -q -c 'ed -mca' > mca.log
  ```

`hpvmcollect`

- The `hpvmcollect` command on the VM Host or on the guest to collect Integrity VM information that is useful in analyzing system problems.
- The `hpvmcollect` command collects log files, system status, device information, system and Integrity Virtual Machines configuration, guest information, and crash dumps.
- By default, the `hpvmcollect` command creates a directory called hpvmcollect_archive in your current directory, and copies all the Integrity VM and VM Host information.
  For example, to gather information for a guest named OVMSG1 on the VM Host, enter the following command:
  ```
  # hpvmcollect -P OVMSG1
  ```
  This command creates a directory called hpvmcollect_archive in your current directory (if it does not already exist) and then collects information about the VM Host crash dump. The information is then put into a tar file format (if there is a crash dump) or a tar.gz file format (if there is no crash dump).

  Note:
  Do not modify the guest configuration before running the `hpvmcollect` command.

Guest disappears

- Check the guest status, using the following command:
  ```
  # hpvmstatus -P OVMSG1
  ```
- Check the guest log from the virtual console:
  ```
  vMP> rec -view
  ```
- If any of the following message is displayed, it is an HPVM problem:
  - "Guest punishment"
  - "VMM panic"
  - "Assertion failed"
- If the log says nothing about stopping the virtual machine, look for a tombstone file in the guest machine directory:

```
# ll /var/opt/hpvm/guests/OVMSG1/tombstone
# ll /var/opt/hpvm/guests/OVMSG1/vm.core
```

- Collect the information using the `hpvmcollect` command. Report the information through your support channel.

Troubleshooting tips

- See the *Integrity VM and OpenVMS Release Notes* for known issues and limitations.
- See the *Integrity VM Release Notes* for the latest patch required by the Integrity VM.
- See the /var/adm/syslog/syslog.log file for the Scheduler and virtual switch issues. (Search for the 'hpvmdvr', 'hpvmntdvr', 'hpvmnetd', 'vm_fssagt', 'hpvmmonlogd', and 'hpvmamrd' strings.)
- See the Global Virtual Machine Monitor log file for errors.
  ```
  # tail –f /var/opt/hpvm/common/hpvm_mon_log
  ```
- To change the size of Virtual Machine Monitor log file:
  ```
  - Use the ch_rc command to change the file size:
   # ch_rc –a –p VMMLOGSIZE=4096
  - Kill the monitor log daemon (it respawns):
   # kill -HUP `cat /var/run/hpvmmonlogd.pid`
  (/etc/rc.config.d/hpvmconf VMMLOGSIZE=1024kb by default)
  ```
- See the guest log (/var/opt/hpvm/guests/<guest>/log) and guest console log (/var/opt/hpvm/guests/<guest>/console/conslog) files for errors.
- Collect other logs with the `hpmvoncolse` command for analysis:
  ```
  # hpvmconsole –P OVMSG1 –q –c cl > cons.log
  # hpvmconsole –P OVMSG1 –q –c 'ed –init' > init.log
  # hpvmconsole –P OVMSG1 –q –c 'ed –mca' > mca.log
  # hpvmconsole –P OVMSG1 –q –c 'rec –view' > op.log
  ```
- See the Virtual firmware logging, Forward Progress Log, and System Event Log:
  ```
  # /usr/sbin/diag/contrib/s1view –p 0 –f FPL
  # /usr/sbin/diag/contrib/s1view –p 0 –f SEL
  # hpvmconsole -P OVMSG1
  [OVMSG1] vMP> SL
  ```
- Collect logs using the `hpvmcollect` command.

Debugging and monitoring tools

This section explains about the debugging and monitoring tools.

Fair Share Scheduler (FSS)

The default scheduler logging level is set to level 3. Messages appear in the /var/adm/syslog/syslog.log file.

For detailed scheduler logging, set log level to 7 (extremely verbose) or 6 (usually sufficient). To set the log level to 7, use the following command:

```
# vm_fssagt -l 7
```

Message description for FSS are given in the following table:

Message	Description
agent state changing to ACTIVE	Entitlements can be achieved
agent state changing to STRESS	Entitlement cannot be achieved (insufficient CPU resources, physical CPU failure). If this message continues, you must recheck the guest entitlements
detected significant system_group activity	Scheduler detects significant CPU consumption by VM Host OS. This is not a problem. The System Group contains Host HP-UX operating system, Host daemons, and pseudo processes like swapper and vxfsd.
HPVM scheduler now entering ACTIVELY MANAGED state	Scheduler has recently scheduled a virtual CPU (vCPU)
HPVM scheduler now entering PASSIVE state	Scheduler has not scheduled a vCPU within the last tick

If FSS is constantly in a STRESS mode, it indicates too much system load. Occasional STRESS messages in the Scheduler logging are OK. Check the state of the Scheduler using the following command:

```
# hpvmstatus -S
```

If the state is down, use the following command to collect the logs:

```
# grep vm_fssagt /var/adm/syslog/syslog.log
```

System performance monitoring

Use the following commands and tools to view the system's performance:

- `hpvmstatus`
- `hpvmsar`
- `ps and top`
- `glance/glanceplus`

`hpvmstatus`

- The `hpvmstatus` command displays information about the operational state and the virtual hardware configuration of the virtual machines on the VM Host.

- The `hpvmstatus -s` command gives the information about the resource utilization within the Host.

  The important options available with the `hpvmstatus` command are given in the following table:

Option	Displays
-e	Displays the event log for the VM Host or the specified virtual machine. The event log records all the configuration changes of a virtual machine.
-i	Prints statistics collected by the monitor. Currently, these include vCPU percentage and durations over the lifetime of the guest.
-r	Displays the CPU entitlement information for the virtual machines.
-d	Displays the devices on the specified virtual machine.
-V	Displays detailed information (verbose mode) about the virtual machines.
-s	Displays server system resources.
-S	Displays the mode the schedule is in.

`hpvmsar`

- The `hpvmsar` command is similar to the Monitor utility on an OpenVMS. The `hpvmsar` command displays the guest CPU utilization with statistics like Busy time, Idle time, Wait time and Host time.
  The field descriptions are given in the following table:

Field name	Field description
Busy time	Counts whenever the guest runs.
Idle time	Counts when the guest has no activity and the CPU is returned to the Host.
Wait time	Counts when the guest is pre-empted, but has some activity pending.
Host time	Counts when the Host is running, as seen from the guest's point of view.

- With the -a option, the `hpvmsar` command displays information about all the running guests.
- With the -A option, the `hpvmsar` command displays information about all the guests, whether they are running or not.
- Collecting data: Use the following command to collect data for 2 seconds for all the running guest:
  ```
  # hpvmsar -s 2 -a
  ```

`ps and top`

- Each VM is manifested as a UNIX process running on the VM Host, the physical resources including CPU, I/O, and so on, consumed by a given VM, can be identified by monitoring the process associated with that VM.
- These processes have the executable name hpvmapp and typically have the option –d whose argument name is the name of the VM.
  For example, the process with the command `'hpvmapp -d OVMSG1'` corresponds to the virtual machine named 'OVMSG1.'

- Tools such as the ps and top can be used on the VM Host to monitor a virtual machine by identifying the Process ID (PID) for a given VM.
  For example, the PID for some VM can be identified from the output of '`ps –fu root | grep hpvmapp`' and then used with top to identify the resources being consumed by that VM.

`glance/GlancePlus`

- The glance tool gives information about the system utilization including CPU, memory, disk and so on.
- It consists of two components:
  - Motif-based program - "gpm"
  - Character mode program -"glance"
- You can use the `glance` command on the Host:
  `# glance`

  Note:
  The glance tool will automatically start and update the data at fixed frequency.

- glance –V provides configured and current entitlements, logical and physical CPU usage, and guest uptime for all the guests.

The important options available with the glance tool are given in the following table:

Option	Displays
A	All CPUs detail report
C	CPU detail report
D	Disk detail report
L	LAN detail report
M	Memory detail report
V	Logical system detail report
h or f7	Online help
e or f8	Exit

Device management

`hpvmdevinfo` and `hpvmdevmgmt`

- To display the devices information and manage devices, use the `hpvmdevinfo` and `hpvmdevmgmt` commands.
- The `hpvmdevinfo` command displays the information about the storage devices assigned to a virtual machine.
- Use the `hpvmdevinfo` command to find mappings between the virtual disks and the physical disks.

- Use the `hpvmdevmgmt` command to manage the devices that are associated with the VM Host and the guests.
- Check the device database. Use the following command to see if the virtual devices you created are listed. If they are not listed, you must create them again.
  ```
  # hpvmdevmgmt -l all
  ```
- To check all the devices associated with the particular guest (OVMSG1), use the following command:
  ```
  # hpvmdevmgmt -l gdev:depend:OVMSG1
  ```
- Restricted devices cannot be associated with any guest. To check all the restricted devices, use the following command:
  ```
  # hpvmdevmgmt -l rdev
  ```
- To check all the guest devices, use the following command:
  ```
  # hpvmdevmgmt -l gdev
  ```

Error logging

`dmesg`

- To collect system diagnostic messages to form error log.
- To verify if the hpvmdvr and hpvmntdvr drivers are loaded with no errors:
  - The hpvmdvr driver is used for communicating between the HPVM components.
  - The hpvmntdvr driver is used for communicating between the HPVM networking components and the guest.
- The `dmesg` command is used with cron to produce the error log /var/adm/messages. Use the following command:
  ```
  # /usr/sbin/dmesg - >> /var/adm/message
  ```
  Every 10 minutes

References

- HP Integrity Virtual Machines 4.2: Installation, Configuration, and Administration guide: http://bizsupport1.austin.hp.com/bc/docs/support/SupportManual/c02023903/c02023903.pdf

- HP Integrity Virtual Machine 4.2.5: Release notes: http://bizsupport2.austin.hp.com/bc/docs/support/SupportManual/c02545643/c02545643.pdf

- HP OpenVMS Version 8.4 Upgrade and Installation Manual: http://h71000.www7.hp.com/doc/84final/ba322_90087/ba322_90087.pdf

- HP OpenVMS Version 8.4 New Features and Documentation Overview: http://h71000.www7.hp.com/doc/84final/6679/ba322_90088.pdf

- HP OpenVMS Version 8.4 Release Notes: http://h71000.www7.hp.com/doc/84final/6677/ba322_90089.pdf

OpenVMS Technical Journal V16

Data Encryption using Archive Backup System

Nagendra Shastry

About this document

This document provides information about the following topics:

- Using software encryption with the Archive Backup System (ABS)
- Using hardware encryption with the ABS
- Best practices and limitations

Intended audience

This document is intended for:

- Administrators who have implemented the ABS backup solution in their environment.
- Administrators who need a software or hardware based encryption solution in their ABS environment.

Prerequisite for using this document

- Knowledge of ABS and MDMS objects (Refer: ABS Operation Guide and Reference Guide).
- Knowledge of taking backup using the ABS saves and restoring data using the ABS restore operations (Refer: ABS Operation Guide).

Introduction

Encryption overview

Encryption is derived from the Greek word kryptós, which means hidden. It is the process of concealing information from unauthorized parties by means of a mathematical cipher. It is an algorithm that disguises the underlying text unless the reader has the de-cipher code. In encryption technology, the cipher is a complex mathematical algorithm that is applied to the unencrypted data, also known as "plain text", to produce encrypted data known as "cipher text".

A simple example of using a cipher is alphabetical substitution such as replacing each letter of the alphabet with a different letter. For example, A=D, B=I, C=J, and D=Z. Unfortunately this method is very limited, as a quick analysis of the letter frequency would easily reveal the substitution code used. More complex ciphers change the substitution each time it is used. For example, AAAA is encrypted as DFGT or similar random set of characters instead of HHHH, the simplified A=H substitution. The complex class of cipher is known as poly-alphabetical. Clearly far more complex mathematical algorithm ciphers are used today to ensure that the code cannot be broken by force.

To decrypt a message, a key is required and the correct mathematical algorithm. When the key is changed, the cipher completely alters the substitution sequence used. The correct key is required for recovering the original plain text.

Encryption is a way to make data unreadable to others while still allowing authorized users to access it. Encryption requires the user or system to have a specific key and software to encrypt and

decrypt the data. Encryption uses various mathematical algorithms for transforming plain text into cipher text and back again. The strength of the encryption depends on the type of algorithm and the key used to encrypt the data.

There are two types of algorithm:
- Defense Encryption Standard (DES)
- Advanced Encryption Standard (AES)

For more information about the DES and AES supported algorithm, see the section "Support matrix" of this document.

Encryption in ABS

This section provides information about the supported encryption types in ABS. It also provides information on how the key management facility is used to generate and maintain the used key for software-based encryption and hardware-based encryption.

ABS Version V4.5 and later supports the following types of encryption:
- Software encryption
- Hardware encryption

Note:
The key management facility is supported only on software encryption. For hardware encryption, you need to use the supported hardware for maintaining the key.

Software encryption

Software encryption is a method where the data is encrypted before leaving a server for writing on the targeted storage medium such as a disk or a tape. In ABS, data is encrypted by underlying the BACKUP utility. Since software encryption happens in the host system, ABS requires extra CPU cycles apart from reading and writing the data.

ABS supports data encryption in the following ways:
- Encrypting data without key management
- Encrypting data with key management

How to use software encryption without key management?

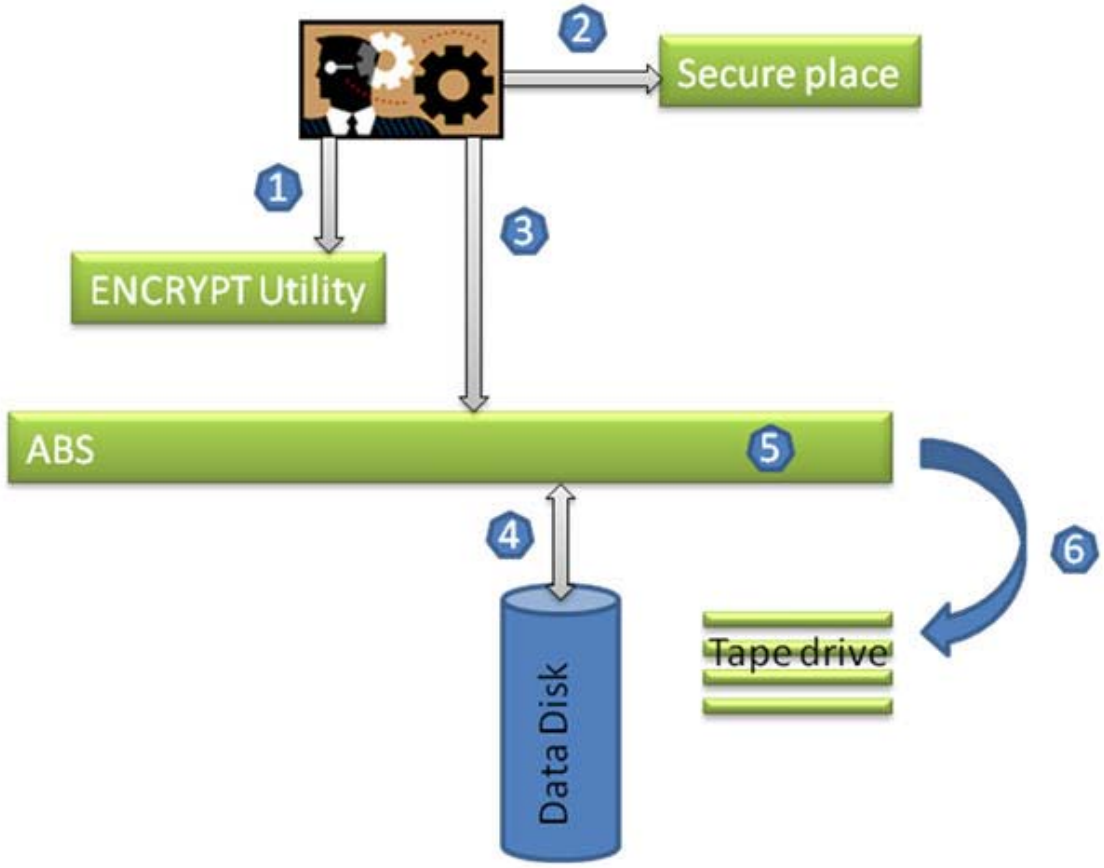
In this method, you have the flexibility of creating a key and passing the key to ABS to encrypt the data, using the required encryption algorithm. The key used for encryption has to be stored securely by you for future restore operation. ABS does not maintain encryption key. If you lose the key, ABS will not be able to restore the data. If you choose to pass the key name or key value, then the key must be created using the ENCRYPT utility on an OpenVMS node/host system where the

key is being used with the ABS. Alternatively, you can pass the key value directly to the ABS without creating the key name using the ENCRYPT utility.

Note:

HP recommends using key name to pass a key value to ABS for encryption instead of directly passing the key value to ABS, because the key value gets displayed in the ABS save/restore log file.

The following figure demonstrates how encryption happens without key management.



1. Generates key by using the ENCRYPT utility.
2. Stores generated key in a secure place.
3. Passes the generated key to ABS.
4. Reads the data from the Data Disk.
5. Encrypts the data using the key passed by the user.
6. Passes the encrypted data to tape drive for writing on to the tape.

The following example explains how to use ABS to encrypt the data without key management. There are two methods of encrypting data without using the ABS key management. In both the methods you need to manually pass the key or key name to ABS.

1. Specify the key to use for encrypting in the selection object. This makes the key visible to any user.
   Or
2. Create a key and associate a name to that key so that the key name can be used to specify in the selection object.

Note:
HP recommends using the second method because this is more secure to create a key. Hence, only the second method is explained in this document.

The following example explains how to create a key and associate the key with a key name, and how to use the key name to encrypt the data using ABS.

1. Create an encryption key and associate a key name to the key.
   ```
   $ ENCRYPT/CREATE_KEY/PROCESS DATADISK123_KEY_NAME  01234567890
   ```
   This commands provides you an example of how the ENCRYPT utility is used to create a key by assigning a key name for encrypting the data by hiding the key value. You can create the key name either in the JOB, PROCESS, or SYSTEM logical table depending on your security and backup policy. Specify the following qualifiers to control the key name:
   - /JOB The key name is created in the JOB wide table.
   - /PROCESS The key name is created in the PROCESS wide table.
   - /SYSTEM The key name is created in the SYSTEM wide table.
2. Specify the key name DATADISK123_KEY_NAME in selection object along with the algorithm to be used to encrypt the data using the key.
   ```
            Selection: ENCRYPTION
          Description:
       Access Control: NONE
                Owner: TEST_NODE::TEST
     Agent Qualifiers:/ENCRYPT=(NAME=DATADISK123_KEY_NAME, ALGO=DESCBC)
          Before Date: NONE
     Conflict Options: RETAIN_VERSION
     Data Select Type: VMS_FILES
            Date Type: MODIFIED
              Exclude:
              Include:
           Since Date: NONE
          Source Node:
   ```
3. Use this modified selection object to run the save or restore operation.

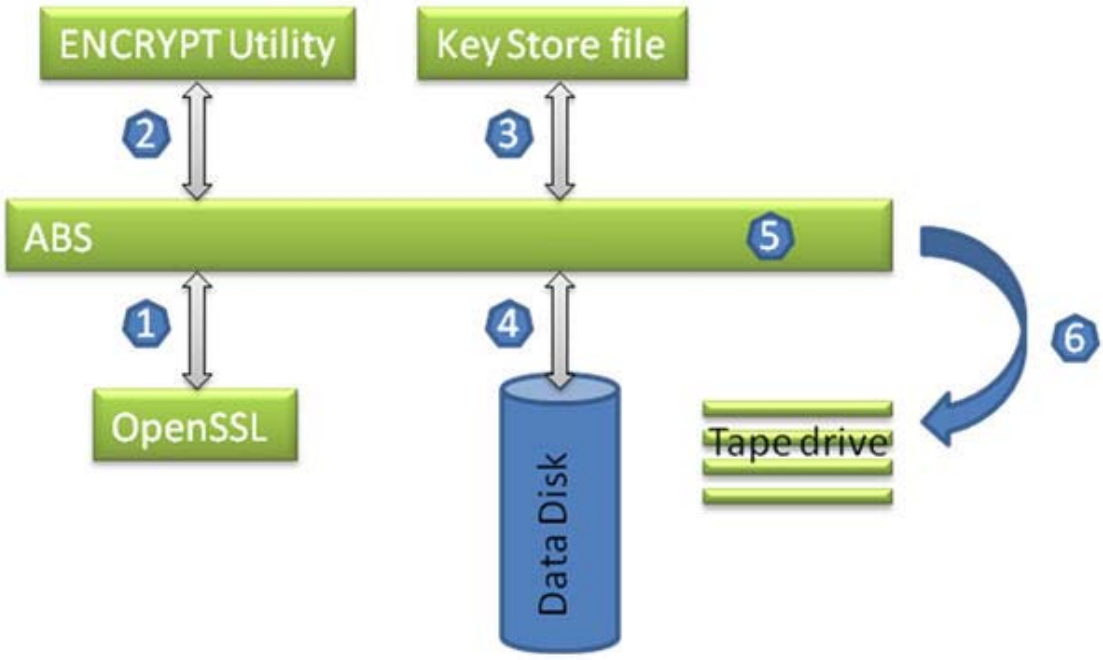
Note:
In this method you have to maintain a key in a secure place for future restore operation. ABS does not maintain the key in its database. If a key is lost, then the data cannot be restored.

How to use software encryption with key management?

It is difficult to keep track of the all the keys used and to create keys manually for every save and associating the key for restore operation is cumbersome. ABS provides an interface to automatically encrypt the data. This interface provides the following functions:

- Generating a key.
- Using the generated key for save operation without user intervention.
- Storing the key in a secure place for future restore operation.
- Retrieving correct key from the key file while restoring encrypted data.

The following figure demonstrates how encryption happens with key management.



1. Makes request to OpenSSL to generate a unique key for encryption.
2. Passes a unique random number to the ENCRYPT utility to create a key for encrypt operation and assign a key name to the key.
3. Stores the generated unique random number in the *key store file*.
4. Reads data from the Data Disk.
5. Encrypts the data using the key.
6. Passes the encrypted data to the Tape drive for writing on to the tape.

The *key store file* name ABS$ENCRYPTION_<node_name>.DAT. <node_name>, is the name of the node/host where the ABS is installed. This file is placed in the ABS$SYSTEM directory.

Steps for running save operation

1. Define a logical "ABS_MDMS_ENCRY_ALGO" with encryption algorithm name for encrypting a data.
   ```
   $ DEFINE/SYS ABS_MDMS_ENCRY_ALGO DESCBC
   ```
2. Include the following line in the save prologue, to generate and store the key:
   ```
   $ RUN SYS$SYSTEM:ABS$ENCRYPT_SAVE.EXE
   ```
3. Modify the selection object to include the /ENCRYPT qualifier with a key name and the algorithm to use for encryption, as given in the following:
   ```
            Selection: ENCRYPTION
          Description:
       Access Control: NONE
                Owner: XION::NAGENDRA
     Agent Qualifiers:/ENCRYPT=(NAME=ABS_MDMS_KEY,
   ALGO='ABS_MDMS_ENCRY_ALGO_TYPE)
          Before Date: NONE
     Conflict Options: RETAIN_VERSION
     Data Select Type: VMS_FILES
            Date Type: MODIFIED
              Exclude:
              Include:
           Since Date: NONE
          Source Node:
   ```

Note:
" 'ABS_MDMS_ENCRY_ALGO_TYPE" is a symbol and not a logical name. Do not forget to include the character " ' " in the symbol.

4. Ensure that this modified selection is specified in the save object before running a save.
5. Run the save operation.

Steps for running restore operation

Note:
For restoring an encrypted data, you do not need to define a logical "ABS_MDMS_ENCRY_ALGO". This is taken from the *key store file* along with the key.

1. Include the following line in the restore prologue to retrieve the required decryption key from the *key store file*:
   ```
   $ RUN SYS$SYSTEM:ABS$ENCRYPT_RESTORE.EXE
   ```
2. Modify the selection object and include the /ENCRYPT qualifier with a key name and the algorithm to use for encryption.
   ```
            Selection: ENCRYPTION
          Description:
       Access Control: NONE
                Owner: XION::NAGENDRA
   ```

```
  Agent Qualifiers:/ENCRYPT=(NAME=ABS_MDMS_KEY,
ALGO='ABS_MDMS_ENCRY_ALGO_TYPE)
       Before Date: NONE
  Conflict Options: RETAIN_VERSION
  Data Select Type: VMS_FILES
         Date Type: MODIFIED
           Exclude:
           Include:
        Since Date: NONE
        Source Node:
```

3. Ensure that this selection is specified in the restore object before running a restore operation.

Hardware encryption

Like software encryption, hardware encryption can also be used to encrypt data that is stored on a tape medium to secure the data. Unlike software encryption, the data in the hardware encryption is encrypted on a tape drive. Hence, the encryption process overhead will not be there in the host processor and the server CPU cycles can be used for other processes.

Prerequisites for using hardware encryption with ABS

- Tape drive and cartridge must be LTO-4
- ABS supports the following hardware as key generator and manager:
  - Secure Key Manager (SKM)
  - Product Level Key (PLK)

Note:
In hardware encryption method, the encryption happens in the LTO-4 tape drive. ABS does not know that the data is encrypted and hence, ABS does not maintain the key used for encryption. The SKM/PLK is responsible for storing and retrieving a key. If a key is lost, then ABS will not be able to restore the data.

An overview of the Linear Tape Open-4

Linear Tape-Open (LTO) is a magnetic tape technology developed as an open alternative to the proprietary Digital Linear Tape (DLT). LTO-4 is the fourth generation of standardization for tape drive and tape media.

LTO-4 tape drive features

The LTO-4 tape drive supports the following features:

- Reads and writes data to an LTO–4 tape medium.
- Encrypts data using the 256-bit AES-GCM mode or drive-based encryption, which is currently supported only on an LTO–4 tape medium.
- Reads and writes data to an LTO–3 medium without encryption.
- Reads data from an LTO–2 medium.

For more information on the LTO-4 tape drive features, see
http://h71028.www7.hp.com/ERC/downloads/4AA1-4878ENW.pdf.

LTO-4 tape medium feature

The LTO-4 tape medium supports 800 GB of native capacity. You can store data in the range of 800 GB to 1600 GB (1:2 compressions) by compressing the data before it is stored on the tape medium.

An overview of the Secure Key Manager

Secure Key Manager (SKM) is a key generator with a secure and centralized encryption key management solution for HP LTO4 enterprise-tape libraries. Using SKM, the key generation and management can be automated according to the security policies for multiple libraries. All the process in SKM occurs transparent to ISV backup applications. SKM provides reliable lifetime key archival with automatic multi-site key replication, high availability, clustering, and failover capabilities.

SKM features

The SKM supports the following features:

- Centralized encryption key management for HP LTO4 enterprise tape libraries:
  - Automatic policy-based key generation and management supporting key/cartridge granularity
  - ISV transparent key archival and retrieval for multiple libraries
  - Extensible to emerging open standards
- Strong auditable security for encryption keys:
  - Hardened server appliance
  - Secure identity-based access, administration, and logging
  - Designed for FIPS 140-2 validation
- Reliable lifetime key archival:
  - Automatic multi-site clustering, key replication, and failover
  - Comprehensive backup and restore functionality for keys
  - Redundant device components and active alerts

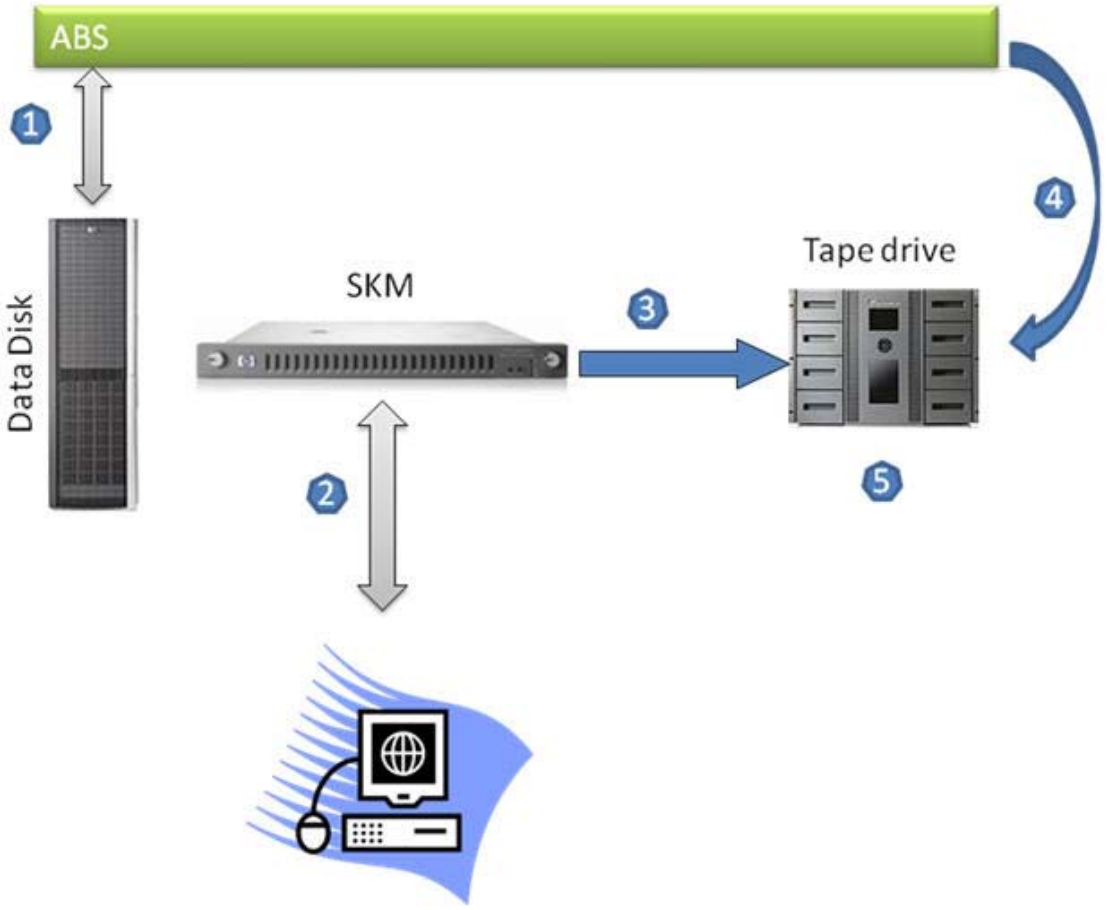
SKM encryption kit requirement

- Compatible Tape Library (CTL) with supported library firmware
- LTO-4 tape drive with supported firmware

Note:
Configuration of SKM and how to use SKM according to the security policy is beyond the scope of this document. For more information, see the *SKM User Guide*.

Hardware encryption using the SKM and LTO-4 tape drive

The following figure demonstrates how hardware encryption happens with LTO-4 and SKM.



1. ABS reads data from the Data Disk for backup operation.
2. Configures SKM for key generation according to the security policy.
3. SKM passes the unique key generated to LTO-4 on request for encrypting data passed by ABS before writing the data on an LTO-4 tape media. SKM stores the key used for encryption for future restore operation.
4. ABS passes data to LTO-4 tape drive in library for writing on to LTO-4 tape cartridge.
5. LTO-4 tape drive first encrypts the data received from ABS using the key passed by SKM and then writes encrypted data on LTO-4 cartridge.

Product Level Key (PLK)

Product Level Key (PLK) is a USB device, which is capable of generating and storing a key used for encrypting data on the LTO-4 tape drive. PLK provides an encryption solution to MSL libraries and Autoloader.

PLK features

The PLK supports the following features:

- Provides access to encrypt and decrypt capabilities of LTO-4 drives.
- Provides secure generation and storage for up to 100 LTO-4 encryption keys on a token.
- Uses random number generator, strong password authentication, and digital envelopes.
- Provides a method for backing up, restoring, and transferring keys.

Operational overview

- With the PLK Encryption Kit, keys are stored on a key server token and are protected by a password (the token's PIN).
- Only one encryption key is used on a cartridge.
- To write encrypted data, you must have the key server token and the password for the key server token.
- If the cartridge contains previously-encrypted data, the key server token with the key for the tape must be in the Autoloader or library.
- To read encrypted data, you must have the key server token with the key for the tape and the password for the key server token.
- Neither the key server token nor the cartridge tapes track the association between the encryption key and the tape. Therefore, it is important to know which token was used with each tape.

PLK Encryption Kit requirement

- A CTL or Autoloader with supported library/autoloader firmware.
- A Tape Library or Autoloader with at least one LTO-4 tape drive with supported firmware.
- An accessible USB port on the back of the Tape Library or Autoloader.

Hardware encryption using the PLK and LTO-4 tape drive

PLK is very similar to USB thumb drive, with key generation and storing capabilities.

The following figure demonstrates how hardware encryption happens with LTO-4 tape drive and PLK.

1. Insert the PLK to USB port of library/loader.
2. Specify the policy to create key in the library/loader console.
3. ABS reads data from the Data Disk.
4. ABS passes data to the Tape drive to store on tape.
5. Tape drive in library/loader encrypts the data received from ABS, using the generated key.

Note:

For more information on how to use PLK and supported hardware/firmware, see the *PLK User Manual*. These topics are outside the scope of this document.

Limitations

- In software encryption using ABS, the key management is not centralized; separate *key store file* is maintained per ABS/MDMS client, including server.
- You cannot decrypt the data if the key used for encryption is not available in the *key store file* where the data restore operation is taking place. In this case, the *key store file* must be copied to a node where you want to restore the encrypted data.
- ABS generates a unique key for each saveset. ABS does not support other type of key generation policy like key per cartridge or key per library and so on.
- Software encryption reduces the effectiveness of drive-based compression.

Support matrix

The following table provides the algorithms supported on an OpenVMS version.

OpenVMS version	Hardware platform	Supported algorithms
OpenVMS 7.3-2	Alpha	DESCBC, DESECB, DESCFB
OpenVMS 8.2/8.3/8.4	Alpha	DESCBC, DESECB, DESCFB, AESCBC128,AESCBC196,AESCBC256,AESECB128, AESECB196,AESECB256,AESCFB128,AESCFB196, AESCFB256,AESOFB128,AESOFB196, AESOFB256
OpenVMS 8.2-1 /8.3/8.3-1h1/8.4	Integrity servers	DESCBC, DESECB, DESCFB, AESCBC128,AESCBC196,AESCBC256,AESECB128, AESECB196,AESECB256,AESCFB128,AESCFB196, AESCFB256,AESOFB128,AESOFB196, AESOFB256

SSL version support

The OpenVMS System must have the compatible SSL version that is compiled with ABS. The ABS V4.5-1200 supports SSL V1.3 and ABS V4.5-1201 supports SSL V1.4. See the *ABS Release Notes* for supported SSL version with a particular ABS version.

Encryption support

Software encryption is supported from ABS V4.5-1200 onwards, and hardware encryption is supported from ABS V4.4A-1001 onwards. See corresponding version of ABS documents to get supported hardware list.

Best practices

Software versus hardware encryption

- Avoid using both software and hardware encryption for the same data. This increases the security, but it has performance overhead on software side.
- Drive-based hardware encryption is best in a situation where software and hardware encryption is available. Encryption of data happens on tape drive, which improves performance and requires no host CPU cycles to encrypt.

Securing *key store file*

For software encryption

Back up the *key store file* regularly to a secure place for successful restore operation. If a key is lost, the ABS and HP cannot help in restoring the encrypted data.

For hardware encryption

Automatic multi-site clustering, key replication, and failover feature of SKM are used to synchronize two SKM. At any time, if one SKM is not accessible, then the key can be accessed from the other SKM. For more information, see the *SKM Configuration Guide*.

PLK supports backing up of keys to a file that can be copied to another library for restore operation. HP recommends taking backup of keys to a file for security reason. For more information, see the *PLK User Manual*.

Handling backup and restore across multiple clients (hosts)

The key generated in each node/host for encrypt operation is placed in the same host in the *key store file*, for restoring encrypted data from different node/host. Hence the *key store file* has to be copied to a node/host, where the restore operation has to be performed; this will enable ABS to get the correct key for restoring encrypted data. The *key store file* must be placed in the ABS$SYSTEM directory.

Troubleshooting

The following table provides the list of errors while using the Encryption feature with key management.

Error message	Possible cause	Recommended steps
Error: Memory allocation failed!	There is no sufficient physical memory to reserve space for encryption key generation/store/retrieve operation	Stop some applications so that encryption process can get free memory for its operation
Error: Encryption Key define failed!	Some error occurred while defining key used for encryption operation	Make sure that the ENCRYPT utility is started and running

Error: Failed to open key file!	Encountered problem while trying to open the *key store file* for the read or write operation	Make sure that the *key store file* is not deleted or moved to different location or renamed
Error: Failed to connect key file!	Not able to open a channel for the read or write operation	Check the process quota and the number of channel it can open
Error: RMS Put operation failed!	Failed to write key into the *key store file*	Check if there is sufficient space in the disk where the *key store file* is placed
Error: Failed to close the key file!	Problem encountered while closing an open *key store file*	Check if any save/restore operation is hanging, if yes, stop jobs and start the save or restore operation
Error: Algorithm logical translation failed!	Encountered problem while translation of algorithm logical	Check that the logical ABS_MDMS_ENCRY_ALGO is defined properly
Error: Failed to translate ABS Object number!	Failed to get thread number of the save or restore operation	Some error for unknown problem. Report to Engineering after collecting log file of the save or restore operation, and save, restore, environment, archive objects information
Error: Failed to translate saveset name!	Failed to get saveset number while running the save or restore operation	Some error for unknown problem. Report to Engineering after collecting the save or restore operation failed log file, save, restore, environment, archive object output
Error: Failed to translate saveset location!	Failed to recognize saveset location on a disk or tape name	Some error for unknown problem. Report to Engineering after collecting the save or restore operation failed log file, save, restore, environment, archive

		object output
Error: OpenSSL API Failed!	Problem while establishing communication with SSL	Check whether Open SSL is started and running
Error: SYS$DEQ Failed!	Not able to unlock the *key store file*	Report engineering with the save or restore log file. Try to take process dump of running save or restore job
Error: LIB$SET_SYMBOL Failed!	Encountered problem while defining symbol to pass encryption algorithm	Report Engineering with the save or restore log file. Try to take process dump of running save or restore job
Error: Failed to translate SYS$NODE logical!	Not able to get a node name	Make sure that SYS$NODE is defined in the system wide table in failed node
Error: SYS$PARSE Failed!	Encountered an error while accessing *key store files* while executing restore operation	Make sure that the required *key store file* is present in the ABS$SYSTEM directory. If the problem is still present, then take process dump of restore job and pass to Engineering
Error: SYS$ENQW Failed!	Encountered problem while getting lock on *key store file* for the read write operation	Report to Engineering with the save or restore log file. Collect process dump of the save or restore job

Frequently asked questions

Can I execute BACKUP/LIST to list content of saveset?
No, ABS encrypts header along with the data. Decryption key has to be supplied along with this command.

What are the advantages of using AES over DES?
AES is a cryptographic algorithm that protects sensitive and unclassified information. The National Security Agency (NSA) reviewed all the AES finalists, including Rijndael, and stated that all of them were secure enough for the U.S. Government non-classified data. In June 2003, the U.S.

Government has announced that AES can be used for classified information by stating the following:

"The design and strength of all key lengths of the AES algorithm (i.e., 128, 192 and 256) are sufficient to protect classified information up to the SECRET level. TOP SECRET information will require use of either the 192 or 256 key lengths. The implementation of AES in products intended to protect national security systems and/or information must be reviewed and certified by NSA prior to their acquisition and use."

Is it possible to restore data if the *key store file* is deleted?
No. If the *key store file* is lost you cannot restore data, even HP cannot restore it.

For more information

- Firmware links
  http://www.hp.com/support
- About Security
  http://h71028.www7.hp.com/enterprise/us/en/solutions/storage-security.html
- About Encryption
  www.hp.com/go/dataencryption
- Tape solution
  www.hp.com/go/tape
- ETLA Tape Libraries
  www.hp.com/go/ETLA
- Enterprise Backup Solutions
  www.hp.com/go/EBS
- Advanced Encryption Standard at the website
  http://csrc.nist.gov/archive/aes/rijndael/
- Linear Tape-Open at the website
  http://www.lto.org/newsite/html/news_4_17_06.html
- Encryption Technology for the HP StorageWorks Ultrium LTO4 tape drive white paper at the website
  http://h71028.www7.hp.com/ERC/downloads/4AA1-4878ENW.pdf
- ABS Support Matrix
  http://h71000.www7.hp.com/openvms/storage/smstape_matrix.html

References

- ABS Documents
  http://h71000.www7.hp.com/doc/abs.html

- Secure Key Manager
  http://h18000.www1.hp.com/products/storageworks/secure_key/index.html

- Secure Key Manager White Paper
  http://h20195.www2.hp.com/v2/GetPDF.aspx/4AA2-1403ENW.pdf

- Federal Information Processing Standards Publications. 1996. Federal Information Processing
  http://www.itl.nist.gov/fipspubs