VMS V5.4

Overview

Bernard OURGHANLIAN

SIG VAX/VMS

18 Octobre 1990

Sommaire

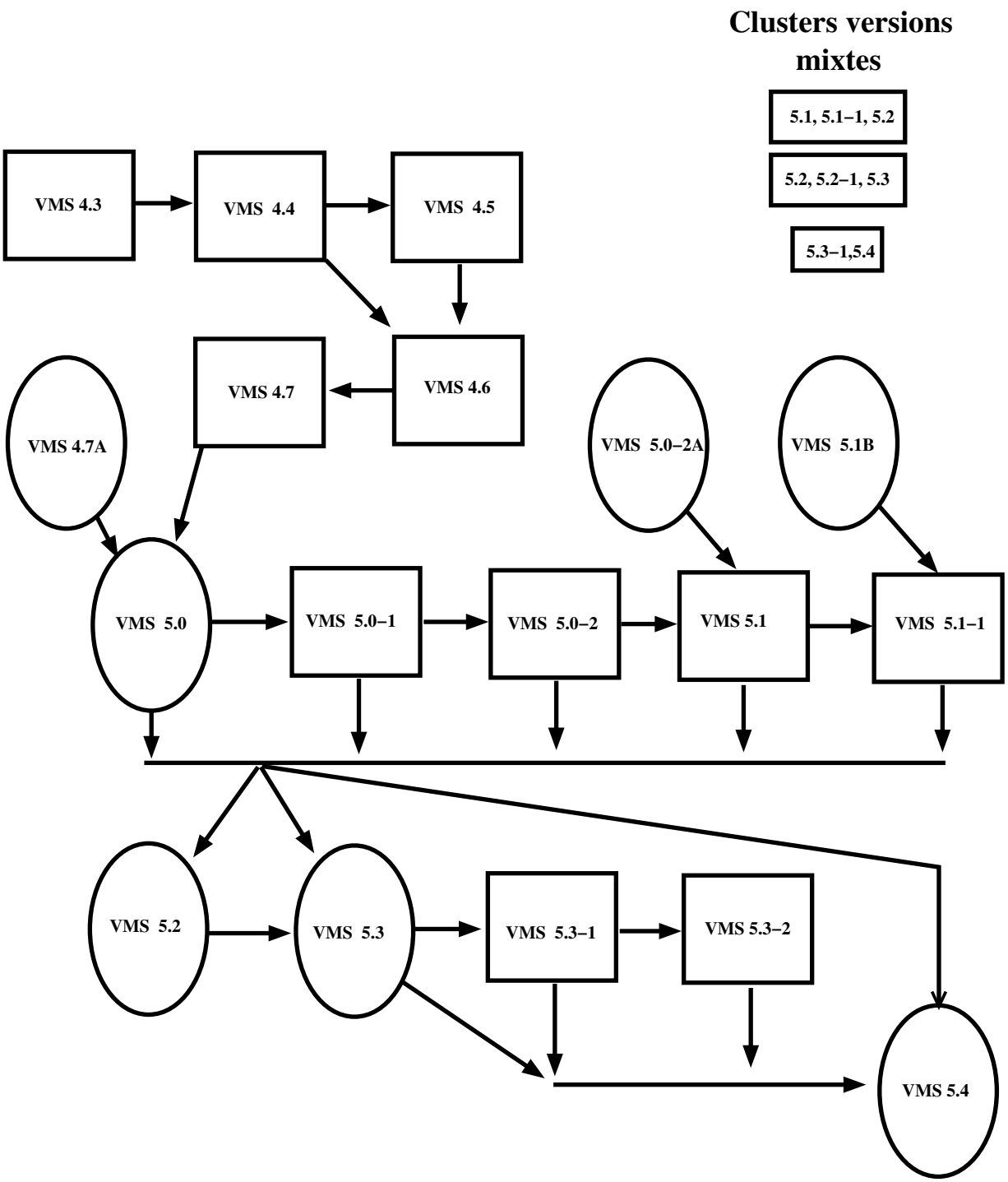
1. Introduction
2. Compatibilité (logiciel et materiel)
3. Installation
4. Support de nouveau hardware et *Primary switching*
5. Nouveautés DCL
6. Nouveautés *system management*

(...)

digital

7. Nouveautés cluster :
    a. *Host Based Volume Shadowing*
    b. *CI load sharing*
    c. *MSCP load balancing*
    d. *Lock Manager*
8. Nouveautés securité
9. Nouveautés RMS
10. *Pseudo terminal*
11. *Display PostScript*
12. Les performances
13. Conclusion

Introduction



Introduction

- Le but de cette présentation est de donner un aperçu des nouvelles fonctionnalités de VMS V5.4

- Pour des raisons de temps, certains sujets ne seront pas abordés :
  -- DECdtm™
  -- certains aspects *d'implementation* ou de fonctionnement interne des nouvelles fonctionnalités
  -- RMS journaling
  -- le support du vectoriel par VMS
  -- ...

Compatibilité

Hardware :

- VMS V5.3 est la dernière version de VMS qui a ete distribuée aux nouveaux clients sur RA60, RL02 et RX33. Il est possible pour les clients ayant un contrat de mise à jour logicielle d'obtenir les kits correspondants sur ces media.
- Il n'est pas supporté d'installer VMS V5.4 sur MicroVAX I, VAXstation I, VAXstation 8000, VAX 11/725.
- Il n'est pas possible d'installer VMS sur un disque système de type RK07, RL02, RC25, RD51 (déjà vrai pour VMS V5.3) et RD52 (nouveau pour VMS V5.4).
- Des disques de taille supérieure à 42 Mbytes sont supportés (sous réserve d'utilisation de VMSTAILOR).

- Un disque système de taille supérieure a 100 Mbytes est recommandé
- Le *standalone backup* n'est pas distribué sur TU58 : on peut installer a partir du *standalone backup* V5.3

Compatibilité

Software :

- Une liste de compatibilité logicielle avec VMS V5.4 se trouve dans le manuel : *VMS installation and upgrade procedure*
- Les logiciels suivants sont concernés par l'installation de VMS V5.4 :
  - -- ALL-IN-1 (V2.4)
  - -- DECdfs (V1.0,V1.1,V1.2)
  - -- RSM server (V2.2)
  - -- VAX FORTRAN (V5.3,V5.4,V5.5)
  - -- VWS
  - -- VAX/VMS services for PCs (V3.1)
  - -- Desktop-VMS

- ALL-IN-1 :
  - -- VMS V5.4 installe 2 images partageables nécessaires pour le support de CDA par ALL-IN-1; ALL-IN-1 étant une image privilegiée, il est nécessaire d'installer ces 2 images (si le support de CDA par ALL-IN-1 est nécessaire)
  - -- Ajouter 2 lignes au fichier de startup dépendant du site d'ALL-IN-1 :
    OA$SITE_BUILD_SHARE:A1V24_SITE_START.COM :

    $INSTALL CREATE SYS$SHARE:XDPS$DPSLIBSHR
    $INSTALL CREATE SYS$SHARE:XDPS$DPSCLIENTSHR
- DECdfs :
  - -- Le kit de VMS V5.4 contient un fichier de mise à jour de DECdfs pour les versions V1.0, V1.1 et V1.2 tournant sous VMS V4.n
  - -- Concerne la correction d'un problème de sécurité
  - -- Fichier SYS$UPDATE: DFSMUP012.A

-- Contenu dans VMS054.D

-- A installer par VMSINSTAL

- RSM :

-- La commande RSM :
MANAGE> INSTALL OPERATING_SYSTEM
peut conduire à un fatal internal RSM error lors de l'installation du 2ème client

-- Ceci provient du fait que, lors d'une commande du type :

$MCR AUTHORIZE ADD/PROXY client::RSM$CMANAGER /DEFAULT

AUTHORIZE retourne correctement un status de type *Warning* qui conduit à l'erreur rapportée par RSM (VMS V5.3 rapportait un status *Informational*)

-- Un contournement consiste à supprimer ce proxy avant chaque nouvelle installation de client :

$MCR AUTHORIZE REM/PROXY client::RSM$CMANAGER /DEFAULT

- FORTRAN :
  -- Si l'on installe VMS V5.4 "from scratch", les images partageables FORRTL et FORRTL2 ne sont pas consistantes avec celles insérées dans IMAGELIB.OLB dans le kit
  -- Il en resulte des problèmes lors de l'installation de FORTRAN (dans l'IVP par exemple) du type :

```
%LINK-I-DATMISMCH, creation date of 16-JUL-1990 09:47 in
          shareable image SYS$COMMON:[SYSLIB]FORRTL.EXE;1
     differs from date of 19-JUN-1990 04:43 in shareable
          image library SYS$COMMON:[SYSLIB]IMAGELIB.OLB;1
%LINK-I-DATMISMCH, creation date of 16-JUL-1990 09:48 in
          shareable image SYS$COMMON:[SYSLIB]FORRTL2.EXE;1
     differs from date of 19-JUN-1990 04:44 in shareable
          image library SYS$COMMON:[SYSLIB]IMAGELIB.OLB;1
```

-- Cela n'est pas grave et peut etre corrigé facilement par une commande du type :

$LIBRARY/SHARE/REPLACE SYS$LIBRARY:IMAGELIB SYS$LIBRARY:FORRTL,FORRTL2

- VWS :
  -- Il est nécessaire de ré-installer VWS après VMS V5.4

- VAX/VMS services for PCs :
  -- Il est nécessaire d'installer la version V3.1 après avoir installé la version V5.4
- Desktop-VMS :
  -- Il n'est pas possible de mettre à jour un système Desktop-VMS par une procédure d'installation standard : il faut attendre la prochaine version de Desktop-VMS

Installation

Les clusters de versions mixtes :

- VMS V5.4 supporte des versions différentes dans le même cluster avec VMS V5.3-1 et VMS V5.3-2
- VMS V5.3 est aussi supporté dans un environnement mixte V5.4 à condition de suivre la procédure contenue dans le fichier :
  SYS$UPDATE:VMS$ROLLING_UPGRADE_054.TXT
  Ce fichier n'est copié sur votre disque système (dans SYS$UPDATE) que si vous répondez *"yes"* aux questions :

  *** Are you performing a rolling upgrade? (Y/N) :**
  *** Are the other system disks running VMS V5.3 or higher ? (Y/N)**

  La procédure de mise à jour vous répond alors :

  If you are running VMS V5.3 (not the same as 5.3-n) on any other system disks while doing this upgrade, a rolling upgrade can not be performed until a fix has been applied.

Is VMS V5.3 running on any other system disks? (Y/N) :

Si l'on répond alors N à la question suivante :

*** Have you applied the required fix to all VMS V5.3 system disks? (Y/N)**

la procédure affiche alors :

************************************NOTICE************************************

This rolling upgrade is not compatible with the 5.3 system.
In order to perform a rolling upgrade with 5.3n, you must take the following action:
1. Read SYS$UPDATE:VMS$ROLLING_UPGRADE_054.TXT on this system disk
2. Perform documented steps
3. Restart upgrade procedures
****

-- Dans ce cas, 2 fichiers :

- MSCP.EXE_V53-1
- VMS$RU_FIXUP_054.COM

sont créés dans SYS$UPDATE et l'on doit copier ces fichiers sur chacun de ses disques système V5.3, appliquer le fix et rebooter (la version intermédiaire ainsi créée est la V5.3-0A)

Le temps d'installation :

- VMS V5.4 demande environ le même temps d'installation que celui nécessaire pour une installation ou mise à jour V5.3: pour la plupart des systèmes entre 1.5 et 4 heures en fonction du médium d'installation et du type du système.
- L'installation de DECwindows, si elle est effectuée séparément au moyen de SYS$UPDATE:DECW$TAILOR.EXE, prend environ 1.5 heures

La place disque nécessaire :

- Pour faire la mise à jour d'un système en V5.4, il est nécessaire de disposer d'au moins 36000 blocks disque libre pour faire la mise à jour (à comparer avec 29000 blocks libres nécessaires pour VMS V5.3)
- Pour installer l'environnement complet VMS, DECwindows avec le support de

tous les langages, il faut 134000 blocks libres, non compris pagefile, swapfile, dumpfile (à comparer avec 110000 blocks necessaires pour VMS V5.3)

- Cet accroissement provient principalement de :
  - -- XDPS (Display Postscript tm): 8000 blocks
  - -- Fichiers exemples DECwindows: 3000 blocks
  - -- Dictionnaire de mots de passe: 1900 blocks
  - -- DECwindows XNLS: 1600 blocks
  - -- VVIEF: 800 blocks

- Tableau de l'espace libre pris en compte par le KITINSTAL :

VMS base kit	79000
>VMS minimum kit	36000
-VMS Required files	30000
-DECwindows required files	6000
>VMS library files	30000
>VMS optional files	13000
DECwindows kit	55000
>DECwindows applications	12000
>DECwindows prog. support	26500
>DECwindows example files	3000
>DECwindows server	8000
>DECwindows 75DPI fonts	2500
>DECwindows 100DPI fonts	3000
Full kit, VMS + DECwindows	134000

- L'expension des *libraries* représente 8500 blocks de plus
- En raison de la place nécessaire, les RD53, RM03, RP05 ne sont plus supportés comme disques système si l'on utilise DECwindows

Avant d'installer :

- Vérifier que le paramètre SHADOWING est à 0 : on ne peut pas faire la mise à jour de VMS sur un disque en *shadow*
- Arrêter DECnet et le *queue manager*
- Avoir un fichier SYSUAF dans SYS$SYSTEM:SYSUAF.DAT
- Avoir un fichier de pagination sur le disque système d'au moins 4600 blocks
- Ne pas avoir un fichier SYS$COMMON:[SYSEXE]SYSDUMP.DMP : pour avoir un fichier de dump commun, il faut l'appeler d'un autre nom : SYSDUMP-COMMON.DMP par exemple
- Vérifier que le paramètre VMS5 est à 0 : ceci est dû au fait que, si l'on est sur une configuration DSSI, il est possible que les noms de disque système et/ou de disque d'installation changent pendant la mise à jour

- Attention au problèmes de vecteurs flottants : le paramètre VMS8 n'a plus de rôle avec VMS V5.4
- En cas de mise à jour de VMS depuis une version antérieure à V5.2 ou en cas de création d'un nouveau disque système, installer le PAK VAXCLUSTER : ceci peut être effectué pendant la procédure d'installation qui invoque VMSLICENCE
- La *release date* de VMS a changé avec VMS V5.4 et devient le 1-JUN-1990 : tout PAK pour VMS ou un SIP précisant une *release date* anterieure au 1-JUN-1990 n'est plus valide
- Dans le cas d'une mise à jour, le PAK VMS est automatiquement mis à jour lors de l'installation de VMS pendant la *mandatory update* (exécution de SYS$UPDATE:LMF$CONFIG.COM, s'il existe)

- ATTENTION de bien disposer d'au moins la version V39B sur les HSCs : les versions suivantes seraient préférables :
  -- V50A sur HSC70/40
  -- V400 sur HSC50

Apres l'installation :

Mandatory Update :

- VMS V5.4 contient une *mandatory update* qui est installée automatiquement pendant l'upgrade (ou l'installation) de VMS V5.4

Fichier à restaurer :

- La procédure d'installation de VMS V5.4 restaure les fichiers système suivants :

  - DECW$SYLOGIN.COM
  - LOGIN.COM
  - RTTLOAD.COM
  - STARTNET.COM
  - SYLOGIN.COM
  - SYSTARTUP_V5.COM
  - SYCONFIG.COM
  - SYPAGSWPFILES.COM
  - SYLOGICALS.COM
  - TFF$STARTUP.COM

- La procédure d'installation "oublie" les 2 fichiers :

  - SYSHUTDWN.COM
  - SYSECURITY.COM

  il faut donc les restaurer : c'est un bug dans STARTUP.MAN fixé en V5.4-1

Messages système et HELP :

- La version V5.4 permet d'obtenir les messages système depuis le HELP
- Ces messages se trouvent dans le fichier SYS$HELP:SYSMSGHELP.HLB
- Ce fichier est en format compressé : il est possible de décompresser le fichier, mais pas par LIBDECOMP : il faut le faire à la main par la commande : $LIBRARY/DATA=EXPAND
- Cela consomme 1600 blocks d'espace disque supplémentaire

Installation V5.4-0A :

- Sur les systèmes de type VAX 9000 et VAX 6000-5xx, il FAUT installer la *mandatory update* V5.4-0A
- Sur les systemes où l'on envisage d'utiliser le *host based volume shadowing* (bien que cela ne soit pas supporté sur des VAXes autres que les VAXft 3000), il FAUT installer la *mandatory update* V5.4-0A

Suppression du démarrage des processes DECdtm :

- Si l'on n'utilise pas DECdtm, il est possible de supprimer son démarrage en définissant le nom logique SYS$DECDTM_INHIBIT dans SYLOGICALS.COM
- Ce nom logique, defini /SYSTEM/EXEC, peut être n'importe quelle chaine de caractères

Support de nouveau hardware

- VMS V5.4 supporte les nouveaux processeurs suivants :
  - -- Les VAX 9000s modèle 210, 410, 420, 430, 440
  - -- Le VAXft 3000 modèle 310
  - -- Les VAX 6000 modèle 510, 520, 530, 540, 550, 560
  - -- L'InfoServer 100
- Les 2 premiers processeurs ayant été déjà traités pendant le symposium, nous ne traiterons que :
  - -- Les VAX 6000 modèle 510, 520, 530, 540, 550, 560
  - -- L'InfoServer 100

Le VAX 6000 modele 5xx

- 4ème membre de la famille des VAX 6000
- 1 a 6 CPUS
- 13 MIPS SPECmark par CPU
- Implémentation CMOS III (5 nouveaux chips : CPU (MP-chip), FPA (MF-chip), *backup cache controler chip* (MC-chip), clock (CLK-chip), *system support chip* (SSSC))
- Mémoire : 512 MBytes mais support d'un adressage physique étendu sur 32 bits
- Clock : 16 ns
- Cache primaire sur le chip : 2 KBytes

- Cache secondaire :
  -- *writeback*
  -- 512 KBytes protégés par ECC

- Bus central XMI 2 : extension du bus XMI supportant un protocole permettant le *writeback* cache
- Jusqu'à 2 processeurs vectoriels (XRV identiques à ceux se trouvant sur VAX 6000 modele 4xx) optionnels : 1 sur 510, 530 et 540, 2 sur 520, 0 sur 550 et 560
- Support de l'InfoServer 100 pour l'installation logicielle

L'InfoServer 100

- L'InfoServer 100 est un serveur de disques virtuels fonctionnant sur la base d'un modèle client-serveur
- Ce n'est pas un serveur de fichiers : l'InfoServer 100 n'impose PAS de *file system* sur les disques qu'il sert : un InfoServer 100 peut servir un disque de n'importe quelle structure : c'est le *host* client qui interprète la structure du disque.
- Plusieurs structures disques peuvent être servies et accédées par le même InfoServer 100 en même temps

Le hardware :

- Basé sur un hardware de MicroVAX 3100 :
  - -- CVAX et CFPA
  - -- 4 Mbytes de mémoire protegée par parité; extensible à 8 Mbytes
  - -- *Time of Year Clock*
  - -- 256 Kbytes de ROM (spécifique Info-Server 100)
  - -- 4 lignes série (dont 1 avec controle modem limité)
  - -- 2 ports SCSI (interne et externe)
  - -- 1 RZ23 interne
- 2 variantes : 1 ou 2 RRD40 interne(s)
- Jusqu'à 6 boitiers d'extension pour le modèle avec 2 CDROMs internes et 7 boitiers d'extension pour le modèle avec 1 CDROM interne

Le software :

- Le noyau logiciel de l'InfoServer 100 s'appelle VXT (Virtual X Terminal)
- *Management* :
  - -- 2 façons : en connectant un terminal sur une ligne série de l'InfoServer 100 ou en utilisant un serveur de terminal LAT (service offert par defaut : LAD_Ethernet_Address)
  - -- mot de passe par défaut : ESS que l'on peut changer par la commande SET SERVER PASSWORD
  - -- commandes principales :

>COPY : copie les données d'une partition
> CREATE : crée une partition (read/write)
> DELETE : supprime une partition
> INITIALIZE : formatte un disque InfoServer (format spécifique InfoServer 100)
> LOOP/MONITOR : répètent une comande
> REBOOT : shutdown et reboot
> RESTORE : restaure une configuration précédemment sauvegardée
> SAVE : sauvegarde une configuration
> SET, SHOW : positionne et montre des paramètres
> UPDATE : mise à jour logicielle de l'InfoServer 100

- Fonctionnalités :
  -- L'InfoServer 100 utilise les protocoles LAD (Local Area Disk) et LAST(Local Area System Transport)
  -- LAST est un protocole de transport orienté transaction qui ne permet aucune fonctionnalité de routing et ne fonctionne que sur un LAN
  -- LAD est un protocole specialisé disque utilisant LAST comme transport qui procure les mécanismes nécessaires pour lire et écrire des blocks disques indépendamment du *file system*
  -- L'InfoServer 100 a une fonctionnalité de type *automount* permettant la reconnaissance immédiate d'un nouveau volume avec la création d'un nouveau service reseau portant le nom du label du disque

-- L'InfoServer 100 est livré avec un disque RZ23 pré-installé permettant de sauvegarder les paramètres et caractéristiques de l'InfoServer 100

-- L'InfoServer 100 offre une fonctionnalité de type *load balancing* statique pour le cas ou plusieurs média identiques sont servis sur le réseau (calcul d'un *rating* de service mis à jour toutes les 10 secondes)

-- L'InfoServer 100 ne procure pas de mécanisme de vérouillage dans le cas d'accès en lecture/ecriture mais plutôt une *connection policy* évitant les conflits d'accès. Il y a 2 règles :

> Par defaut l'InfoServer 100 n'autorise les accès en écriture que lorsque les accès en lecture ont ete disconnectés (il y a donc déconnexion des accès en lecture en conflit avec les accès en ecriture avec bloquage de la reconnexion tant que les écritures ne sont pas terminées)

> La deuxieme regle possible est le bloquage des accès en écriture quand un conflit avec une lecture existe : les connexions en écriture sont mises en attente de la deconnexion volontaire des connexions en lecture

-- L'InfoServer 100 *implemente* un certain nombre de fonctions du protocole MOP :

> Identify Self : l'InfoServer 100 répond aux demandes d' identification MOP et envoie un *broadcast* périodique avec son identification

> Down Line Load : l'InfoServer 100 peut être configuré pour répondre aux demandes de téléchargement. Pour répondre à ces demandes, l'InfoServer 100 cherche dans tous ses disques en lecture/ecriture des partitions MOP speciales et repond. L'InfoServer 100 est capable de répondre à 100 demandes de téléchargement simultanées

A quoi sert l'InfoServer 100 ?

- A l'ISL (*Initial System Load*) de tous les VAXes 6000
- Documentation de l'ISL dans la *cover letter*
- Précautions d'utilisation :
  - -- Pas de *boot* de plusieurs *standalone backup* simultanés : le deuxième est ignoré
  - -- DECnet n'est pas nécessaire pour le fonctionnement de l'InfoServer 100 MAIS si DECnet est utilisé, il FAUT le démarrer AVANT le démarrage de la partie client du logiciel de l'InfoServer 100
  - -- Si DECnet n'est pas utilisé, il faut définir le nom logique ESS$IGNORE_DECNET = TRUE

-- En cas d'utilisation de RSM V2.2 et d'un InfoServer 100, il faut suivre ATTENTIVEMENT la procédure indiquée dans la *cover letter* pour éviter de *crasher* pendant l'installation

- Accès aux distributions de VMS
- Accès à la documentation en ligne
- Bientôt :
  -- Le support de l'ISL de la majorité des VAXes
  -- Avec un X-window terminal (non encore annoncé), possibilité de paginer *remote*

Primary switching

- Le but de cette nouvelle fonctionnalité est de permettre de déselecter le CPU primaire et de basculer les fonctions effectuées par ce CPU sur l'un des autres CPUs (*time keeping*, réponses aux interruptions des péripheriques, traitement des entrées dans la *timer queue*)
- La version V5.4 permet le support sur le VAX 9000 et les VAXes 6000 modèles 65xx du *primary switching*
- Le *primary switching* ne s'applique pas aux systemes asymetriques au niveau hardware (console connectée physiquement à un CPU)
- Le *primary switching* est supporté sur les VAX 6000 modèle 2xx et 3xx mais n'est pas *elabled* (bug au niveau con-

sole qui peut empêcher un reboot après un *primary switch*)

- Le *primary switching* est supporté sur les VAX 6000 modele 4xx mais n'est pas *elabled* (bug VMS fixé en V5.4-1 qui crée un problème pour trouver un nouveau *primary*)
- Moyens d'accomplir un *primary switch* :
  -- La commande $STOP/CPU a ete utilisée
  -- Un *error handler* a demandé l'arrêt du CPU primaire
- Le code d'*error handling* détecte les erreurs sur tous les CPUs :
  -- quand un seuil d'erreur est depassé (3 erreurs d'un type en 3 secondes ou 10 erreurs de n'importe quel type), l'*error handler* appelle une routine pour demander l'arrêt du CPU concerné

-- on vérifie si le CPU est un CPU primaire ou secondaire

-- si c'est un CPU secondaire, on continue

-- si c'est un CPU primaire et que le *primary switching* est autorisé, on effectue le *primary switch* et on continue

-- on envoie un message à la console et à l'errorlog

- Un CPU qui a été arrêté peut être redémarré par la commande START/CPU : si le CPU ainsi arreté était un CPU primaire, il redémarre comme secondaire

Nouveautes DCL

- F$CSID :
  - -- retourne à chaque appel le cluster CSID d'un membre du cluster
  - -- met à jour une variable de contexte
  - -- peut être appelé en boucle pour obtenir tous les CSID des membres d'un cluster. Lors du dernier appel en boucle, retourne une chaine vide
  - -- identique d'utilisation à F$PID
  - -- format : F$CSID(context)
  - -- suivi en general d'un appel à F$GETSYI
- F$DEVICE :
  - -- utilise $DEVICE_SCAN pour retourner les noms de péripheriques dont dispose le système

-- il n'y a *pas d'ordre* (chainage des DDBs)

-- format :
F$DEVICE(devnam,devclass,devtype,stream)

-- on peut utiliser le wildcarding (*,%)

- F$ENVIRONMENT :

-- retourne une information sur l'environnement DCL dans lequel on tourne

-- format : F$ENV(item)

-- ajoute les items :

> DISIMAGE

> RESTRICTED

> VERB_SCOPE (environnement LOCAL ou GLOBAL) pour la substitution des symboles

Nouveautes DCL : SET HOST/DTE

- Possibilité de controle du DTEPAD :
  - -- CTRL/ permet de générer un break
  - -- CTRL\ permet de quitter sa session
  - -- CTRL@ permet de passer en mode commande
  - -- nouveaux *qualifiers* :
    - > /BREAK= permet de spécifier le caractère de génération de break
    - > /COMMAND= permet de spécifier le caractère de passage en mode commande
    - > /ECHO
    - > /EIGHTBIT

> /ERROR_ACTION= EXIT,CONTINUE

> /ESCAPE= permet de specifier le caractère de sortie

> /FLOWCONTROL (XON/XOFF)

> /INIT=file permet de spécifier un fichier de commande d' initialisation des paramètres du DTEPAD. Pointé par un nom logique : DTEPAD$INI

> /LOG=file

> /MAX_BUFFERS= permet de spécifier le nombre maximum de *receive buffers*

> /PARITY=NONE,ODD,EVEN

> /READ_DELAY= permet de spécifier le delai du port modem

> /SPEED=(output,input)

-- En mode commande, on peut utiliser les commandes suivantes :

> CLEAR (déconnection du PAD)

> EXIT (retourne en mode session)

> QUIT (termine la session)

> SAVE (sauvegarde les paramètres de configuration dans un fichier)

> SEND BREAK

> SET DTE modifie les paramètres de configuration (identique aux *qualifiers* de SET HOST/DTE)

> SHOW DTE

> SPAWN

Nouveautes DCL : Link

- Nouveau *qualifier* : /BPAGE [=n]
- Permet de spécifier une taille de page plus importante que 512 : n indique l'exposant d'une puissance de 2 comprise entre 1 et 16.
- Le defaut est 13 (8Kbytes)
- Affecte l'allocation des sections, tourne en general sans problème mais peut consommer plus de mémoire virtuelle

Nouveautes DCL : EVE

- Support d' édition de boites rectangulaires (BOX COPY, BOX CUT, BOX PASTE,...)
- Nouvelle commande : CONVERT TABS
- Nouveaux *qualifiers* :
  - -- /WORK : permet de préciser le fichier de travail permettant le "swap" pour éditer de très gros fichiers ("backing store"). La taille maximum d'un enregistrement dans un buffer passe à 64K-2
  - -- /INTERFACE : permet de spécifier l'utilisation d'un terminal caractère ou d'une station DECwindows

Nouveautes DCL :

Gestion des labels de *subroutines* :

- Il n'est plus possible d'appeler une *sub-routine* dont le point d'entrée est defini dans un autre *subroutine block*
- Exemple non valide :

```
$ CALL BAR
$ MAIN: SUBROUTINE
$
$       BAR: SUBROUTINE
$       ENDSUBROUTINE
$
$ ENDSUBROUTINE
```

- Exemple valide :

```
$ MAIN: SUBROUTINE
$
$       BAR: SUBROUTINE
$       ENDSUBROUTINE
$
$CALL BAR
$ ENDSUBROUTINE
```

- Si le point d'entrée d'une *subroutine* est localisé dans un block IF-THEN-ELSE, il n'est plus possible d'appeler cette routine en dehors de ce block IF-THEN-ELSE
- Exemple non valide :

```
$ IF 1
$ THEN
$       FOO: SUBROUTINE
$       ENDSUBROUTINE
$ ENDIF
$ CALL FOO
```

Remise à jour des dates d'expirations de fichier :

- Certains utilitaires pouvaient mettre à jour les dates d'expiration de fichiers :
  - -- $DIR/FULL
  - -- $DELETE/SINCE, /BEFORE, /BY_OWNER sur les fichiers qui ne sont pas concernés
  - -- $PURGE/SINCE, /BEFORE, /BY_OWNER sur les fichiers qui ne sont pas concernés

-- F$FILE_ATTRIBUTES

- Ce problème est corrigé

Nouveautés system management Autogen :

- Nouveau fichier de rapport du mécanisme de *feedback*
- Validation des noms de paramètres
- Possibilité d'inclure un fichier de paramètres externe à MODPARAMS.DAT
- Possibilité de mieux controler les tailles de fichier de pagination et de swap
- Utilisation possible de MAIL pour s'envoyer automatiquement un rapport
- Génération de noms logiques relatifs au fonctionnement d'AUTOGEN

AGEN$PARAMS.REPORT :

- AGEN$FEEDBACK.REPORT a ete remplacé par AGEN$PARAMS.REPORT
- Le fichier AGEN$PARAMS.REPORT contient maintenant :
  - -- Les informations de *feedback*
  - -- Les informations de type *override*
  - -- Des informations de fonctionnement
  - -- Des *warnings*
- Les informations de *feedback* sont toutes les informations précédemment contenues dans AGEN$FEEDBACK.REPORT

- Exemple :

```
PAGEDYN parameter information:
         Feedback Information.
           Old value was 319489, New value is 319489
           Current paged pool usage: 223312 bytes.
            Paged pool request rate:0 requests per 10sec.
```

- Les informations de type *override* indiquent quand une valeur calculée de

paramètre a ete remplacée par une valeur specifiée dans MODPARAMS.DAT

- Exemple :

```
VIRTUALPAGECNT parameter information:
         Override Information - parameter calculation has been overriden.
            The calculated value was 139072.  The new value is 200000.
            VIRTUALPAGECNT is not allowed to be less than 200000.
```

- Les informations de fonctionnement indiquent comment AUTOGEN a tourné
- Exemple :

```
AUTOGEN Parameter Calculation Report on node: ALPHA
This information was generated at 22-AUG-1990 13:40:27.03
AUTOGEN was run from GETDATA to SETPARAMS using CHECK_FEEDBACK
```

- Les *warnings* indiquent les problèmes rencontrés par AUTOGEN pendant son exécution
- Exemple :

```
**WARNING** - Invalid parameter name: SCNODE
The following record is suspect:
     SCNODE = ALPHA
```

Fichier d'include dans MODPARAMS.DAT :

- Il est maintenant possible de prévoir des fichiers d'inclusion dans MODPARAMS.DAT
- AUTOGEN verifie les problèmes de redondance et d'erreur
- Permet des valeurs de paramètres *clusterwide* et des valeurs de paramètres specifiques
- Format :
  AGEN$INCLUDE_PARAMS <fichier>
- Exemple :
  AGEN$INCLUDE_PARAMS
  SYS$MANAGER:Cluster_MODPARAMS.DAT

Possibilité de préciser Min_, MAX_, ADD_ pour PAGE et SWAP file :

- Il est possible dans MODPARAMS.DAT de modifier la façon dont AUTOGEN traite PAGEFILE et SWAPFILE
- Exemple :

```
MIN_PAGEFILE=75000
MAX_SWAPFILE1=5000
MIN_SWAPFILE2=1250
MIN_SWAPFILE3=1000
```

AGEN$MAIL.COM :

- Pour informer les system managers, AUTOGEN exécute SYS$UPDATE:AGEN$MAIL.COM si le fichier existe
- Cette procedure peut etre écrite par le *system manager*, par exemple, pour s'envoyer le fichier AGEN$PARAMS.REPORT

- Exemple :

```
$MAIL/SUBJ="Parameter report for node ALPHA"-
SYS$SYSTEM:AGEN$PARAMS.REPORT
ALPHA::SYSTEM
```

Définition de symboles de status d'éxécution :

- Quand AUTOGEN a terminé, 3 noms logiques sont definis :
  - -- AGEN$P1 : Phase de départ d'AUTOGEN
  - -- AGEN$P2 : Phase de fin d'AUTOGEN
    - > Si une erreur survient, "_E" est mis en fin de la phase où l'erreur est arrivée
  - -- AGEN$P3 : Mode d'exécution utilisé
    - > FEEDBACK ou NOFEEDBACK
  - -- Exemples :

AGEN$P1:	**SAVPARAMS**	**GETDATA**
AGEN$P2:	**TESTFILES**	**GENPARAMS_E**
AGEN$P3:	**FEEDBACK**	**NOFEEDBACK**

LRPCOUNT est pris en compte par le mécanisme de *feedback* :

- Calcul :
  - -- Si une extension de la liste des LRP a eu lieu, augmenter la valeur de 25% du montant de l'extension. Ceci ne peut pas etre superieur a 1,5 fois la valeur courante

    new=current+(peak-current)/4 (<current*1.5)
  - -- S'il n'y a pas eu d'extension, réduire la valeur de 12.5%
  - -- Exemple :

LRPCOUNT parameter information:
Feedback information.
Old value was 29, New value is 35
Maximum Observed LRP list size : 54

LNMSHASHTBL est pris en compte par le mécanisme de *feedback* :

- Calcul :
  - -- Positionnement à la valeur de la première puissance de 2 plus grande que la moitié du nombre d'entrées dans la table de noms logiques système
  - -- Minimum à 128
  - -- Exemple :

```
LNMSHASHTBL parameter information:
        Feedback information.
          Old value was 128, New value is 256
          Current number of sharable logical names : 500
```

GBLPAGES et GBLSECTIONS :

- Les valeurs utilises lors du feedback sont des valeurs de pic et plus des valeurs instantanées
- Calcul :
  - -- GBLPAGES : la valeur maximum de :
    - > Pic observe + 20% + besoins buffers globaux RMS
    - > Nombre demandé par VMSIMAGES.DAT
    - > 80% de la valeur courante
  - -- GBLSECTIONS : la valeur maximum de :
    - > Pic observe + 20%
    - > Nombre demandé par VMSIMAGES.DAT
    - > 80% de la valeur courante

Nouveautés system management Sysman :

- Support d'un fichier d'initialisation
- Controle de la *verification* DCL
- Possibilité d'exécuter une procédure de commande SYSMAN
- Definition de clé
- Possibilité de SPAWN

Fichier d'initialisation :

- Ce fichier est executé à chaque fois que l'on invoque l'utilitaire SYSMAN
- Le fichier d'initialisation par defaut est le fichier SYS$LOGIN:SYSMANINI.INI
- Il est possible d'utiliser le nom logique SYSMANINI pour pointer vers le fichier d'initialisation
- On peut utiliser cette fonctionnalite pour
  -- définir des clés
  -- définir l'environnement SYSMAN

***Verification* DCL** :

- Il est possible de controler le comportement de la *verification* des prochaines commandes DCL executées par la commande DO en modifiant son profil SYSMAN par la commande SET PROFILE/[NO]VERIFY

Exécution de procedures de commande:

- La commande @ permet l'exécution de commandes SYSMAN
- Exemple du fichier TEST.COM

```
SET ENVIRONMENT/NODE=ALPHA
SHOW PROFILE
$RUN SYS$SYSTEM:SYSMAN
SYSMAN> SHOW ENVIRONMENT
%SYSMAN-I-ENV, current command environment:
                Local node only
SYSMAN> @TEST
%SYSMAN-I-ENV, current command environment:
                Individual nodes: ALPHA
                Username BERNARD    will be used on
                non local nodes

  %SYSMAN-I-DEFDIR, default directory on node
   ALPHA -- USER1:[BERNARD]
   %SYSMAN-I-DEFPRIV, process privileges on node
   ALPHA

   CMKRNL
   CMEXEC
   SYSNAM
    .....
```

Définition de cles :

- Possibilité de définition de clés de la même manière qu'en DCL
- Exemple :

```
$RUN SYS$SYSTEM:SYSMAN
SYSMAN> DEFINE/KEY PERIOD "SHOW -
PROFILE"/TERMINATE/ECHO
SYSMAN>
%SYSMAN-I-ENV, current command environ-
 ment:
        Individual nodes: ALPHA
        Username BERNARD    will be used on
         non local nodes

%SYSMAN-I-DEFDIR, default directory on node
ALPHA -- USER1:[BERNARD]
%SYSMAN-I-DEFPRIV, process privileges on
node     ALPHA
    CMKRNL
    CMEXEC
    SYSNAM
            .....
```

Possibilité de SPAWN :

- SYSMAN supporte SPAWN et ATTACH

Nouveautes system management Error log:

- La version V5.3 d'ERF est capable de traduire les entrées d'erreurs V5.4 (pour les péripheriques déjà existants en V5.3)
- La version V5.4 d'ERF est capable de traduire les entrées d'erreurs de toutes les précédentes versions de VMS (depuis V4.0)
- Les entrées de l'error log comprennent l'*uptime* depuis le dernier reboot :

```
******************************** ENTRY 1. ********************************
ERROR SEQUENCE 0.   LOGGED ON:              SID 0A000004
DATE/TIME 2-JUL-1990 10:16:51.71            SYS_TYPE 04010002
SYSTEM UPTIME: 0 DAYS 00:18:04
SCS NODE: SOS4          VAX/VMS V5.4
```

- Nouveaux *qualifiers* :
  - -- /SID_REGISTER=(...[,...])
    - > Permet plusieurs valeurs (un 8800 a plusieurs CPUs et chacun d'eux a son SID mais un 6xxx a plusieurs CPUs mais un seul SID)
    - > Plus besoin de preciser "%X"
  - -- /NODE_NAME=(....[,...])
    - > Permet de spécifier des entrées provenant d'autres systèmes
    - > Ceci n'indique pas que ERF supporte des erreurs *clusterwide*
  - -- /INCLUDE= ou /EXCLUDE=
    - > Supporte 2 nouveaux mots-clé
      - + VECTORS : erreurs liées aux processeurs vectoriels
      - + INFORMATIONAL : messages d'information MSCP

Nouveautés system management Startup :

- Reordonnancement du STARTUP pour permettre la configuration des périphériques avant l'installation des fichiers page/swap
- Les noms logiques sont maintenant définis APRES la configuration des péripheriques et AVANT l'installation des fichiers de page/swap
- SYCONFIG est exécuté très tôt dans le STARTUP (avant les SY*.COM)

Ancien ordre :

- Phase INITIAL
  - -- Démarrage initial de CONFIGURE
  - -- Installation des fichiers de pagination et de swap **SYPAGSWPFILES.COM**
  - **-- SYLOGICALS.COM**
- Phase CONFIG
  - -- Stop CONFIGURE
  - -- Démarrage initial de DECwindows
  - -- Démarrage des processes servers : ERRFMT, CACHE_SERVER, CLUSTER_SERVER,... **éxécution de SYSECURITY.COM**
  - -- LMF
- Phase SYSFILES (obsolete avec V5.4)
  - **-- SYCONFIG.COM**
  - -- SYSGEN AUTOCONFIGURE ALL
  - -- Démarrage de CONFIGURE

-- Vérification de licences LMF

- Phase BASEENVIRON
  -- Démarrage de SMISERVER
- Phase LPBEGIN
  -- **SYSTARTUP_V5.COM**
  -- Démarrage complet de DECwindows
- Phase LPMAIN
- Phase LPBETA
- Phase END

Nouvel ordre :

- Phase INITIAL
  -- Démarrage initial de CONFIGURE
- Phase DEVICE (nouvelle avec V5.4)
  -- **SYCONFIG.COM**
  -- SYSGEN AUTOCONFIGURE ALL
  -- Installation des fichiers de pagination et de swap

-- Démarrage de CONFIGURE

-- SATELLITE_PAGE.COM, **SYPAGSWP-FILES.COM**

- Phase CONFIG

-- Démarrage initial de DECwindows

-- Démarrage des processes servers : ERRFMT, CACHE_SERVER, CLUSTER_SERVER,... **execution de SYSECURITY.COM**

-- LMF

- Phase BASEENVIRON

-- Démarrage de SMISERVER

-- Vérification de licences LMF

- Phase LPBEGIN

-- SYSTARTUP_V5.COM

-- Démarrage complet de DECwindows

- Phase LPMAIN
- Phase LPBETA
- Phase END

Host Based Volume Shadowing

!!!!!ATTENTION!!!!!

La version VMS V5.4 ne supporte *Volume Shadowing Phase* II QUE sur des disques de la serie RF sur un VAXft 3000 *standalone*

La version V5.4-1 *devrait* supporter *Volume Shadowing Phase* II

- Le *volume shadowing* maintient des copies identiques de données sur différents disques
- Permet d'améliorer :
  -- la disponibilité des données : si un des volumes devient indisponible, les données sont encore disponibles
  -- la fiabilité des données : si un block de données d'un volume est mauvais, les données peuvent etre accédées depuis un autre volume
  -- les performances en lecture : les lectures sont optimisées entre les différents volumes en fonction de la queue d'attente I/O

- Avant VMS V5.4, le *volume shadowing* n'était disponible que sur les configurations disposant de controleurs HSC : pour appartenir à un *shadow set*, les disques devaient etre connectés au même controleur ou être dual-portés entre une paire de HSCs. Appelé **Volume Shadowing Phase I**
- Avec VMS V5.4+, le *volume shadowing* devient independant des controleurs et supporte tous les disques DSA présents et futurs : les disques peuvent être sur des controleurs différents et appartenir au même *shadow set*. Appelé **Volume Shadowing Phase II**

Host Based Volume Shadowing

Overview :

- Nouveau *design* et nouvelle *implementation* du *Volume Shadowing*
- Complètement distribué
- Architecture symetrique
- *Implementation host based* : toutes les fonctions liées au *shadowing* sont effectuées par VMS
- Permet le *shadowing* au niveau volume pour tous les disques ODS-2
- Autorise le *shadowing* pour tous les disques DSA (RA, RF, RD sur QBUS) de facon *clusterwide*
- Assistance optionelle du controleur (FMR)

Host Based Volume Shadowing

Différences avec HSC-*based* shadowing:

- Toutes les opérations sur le shadow set sont distribuées sur tous les *hosts* et sont donc exécutées par tous les noeuds du cluster
- Les membres du *shadow set* sont servis MSCP pour le reste du cluster, pas l'unité virtuelle
- L'unité virtuelle est créée et gerée directement sur tous les noeuds quelque soient les moyens d'interconnexion
- Chaque host communique avec chaque membre du *shadow set*
- Nouveaux noms pour les unités virtuelles : DSAx

- Nouveaux paramètres SYSGEN
- Nouveau *qualifier* pour MOUNT
- Nouveau driver SHDRIVER
- Nouveau process : SHADOW_SERVER permettant de gérer les operations de copie sur un noeud local

Host Based Volume Shadowing

Configurations :

- *Host based shadowing* est supporté sur:
  - -- UDA50,KDA50,KDB50,KDM70
  - -- RQDXx
  - -- HSCxx
  - -- RAxx
  - -- RFxx
  - -- RDxx sur RQDXx
  - -- ESE20
- VMS/HSC *based shadowing* et *host based shadowing* peuvent etre utilisés sur les mêmes noeuds et/ou dans le même cluster **MAIS** les unités virtuelles sont incompatibles : on ne peut pas

ajouter un *host based shadow member* dans un VMS/HSC *based shadow set* pré-existant

- *Host based shadowing* n'est pas supporté sur:
  -- les disques SCSI
  -- les disques RDxx sur MicroVAX 2000
  -- les disques non DSA (disques MASSBUS...)
- Règles pour être membre d'un *host-based shadow set* :
  -- les membres du *shadow set* doivent être du meme type et de la même géomètrie
  -- maximum de 3 membres par *shadow set*
- Les règles de configuration pour le VMS/HSC *based shadowing* n'ont pas changé :

- -- les membres du *shadow set* doivent être du même type et de la même géomètrie
- -- maximum de 3 membres par *shadow set*
- -- maximum de 8 *shadow sets* par HSC ou paire de HSCs
- -- maximum de 16 *shadow set members* par HSC ou paire de HSCs

- Servir MSCP des *shadow sets* :
  - -- Les membres d'un *shadow set* sont servis MSCP, pas les unités virtuelles
  - -- Pour monter un *shadow set,* un *host* doit avoir accès (soit localement, soit à travers *MSCP server*) à chacun des membres
  - -- Un membre dual-porté entre 2 *hosts* doit etre servi par les 2 *hosts* au cas ou l'un des *hosts* deviendrait indisponible

Host Based Volume Shadowing

Parametres SYSGEN :

- SHADOWING (defaut 0) :
  - -- 0 : Pas de shadowing
  - -- 1 : VMS/HSC *based shadowing*
  - -- 2 : *Host based shadowing*
  - -- 3 : Mode compatible
- SHADOW_SYS_DISK (defaut 0) :
  - -- 0 : Le disque système n'est pas un *host based shadow member*
  - -- 1 : Le disque système est un *host based shadow member*
- SHADOW_SYS_UNIT (defaut 0) :
  - -- 0 a 9999 : Numero d'unité virtuelle du disque système

- SHADOW_MAX_COPY (defaut 4, paramètre dynamique) :
  - -- 1 à 42 : Nombre de *copy threads* autorisés à tourner en parallèle sur ce *host*
  - -- paramètre dynamique dont le changement n'affecte QUE les nouvelles copies
- VMSD3 (non documente) :
  - -- Temps d'attente pendant lequel *Volume Shadowing Phase II* attend un membre perdu du *shadow set*
  - -- Dynamique, défaut 5 secondes
  - -- Si un VAX perd son *path* vers un membre d'un *shadow set* , le *shadowing phase II* fait passer les I/Os en *stall* et tente de trouver un autre *path.* Si le VAX trouve un autre *path* vers le membre perdu ET que ce membre n'est pas le seul membre valide du *shadow set,* la *shadowing*

Phase II supprime ce membre du *shadow set* et redémarre les I/Os

-- Si l'on tourne en mode compatible, le *shadowing Phase I* necessite que le paramètre HSC **drive_timeout** soit au moins positionné à 5 secondes : cette valeur est passée aux disques *online* du HSC pour déterminer combien de temps ceux-ci doivent attendre avant de se déclarer *available* au controleur sur l'autre port

-- Le *shadowing phase II* n'est pas concerné par le paramètre **drive_timeout** du HSC; cependant si *shawowing Phase I et II* coexistent sur le même HSC, il est préférable de positionner VMSD3 à la même valeur

-- Pour positionner VMSD3 à une valeur différente de la valeur par défaut, le bit de poids fort doit être positionné et le *byte* de poids faible doit contenir le nombre de secondes à attendre

-- Exemple : 10s, VMSD3=%X8000000A

- Si le paramètre SHADOW_SYS_DISK est à 1, le paramètre SHADOWING doit être à 2 ou 3. Sinon, VMS positionne automatiquement la valeur de SHADOWING pendant le boot :
  -- Si SHADOWING = 0, SHADOWING = 2
  -- Si SHADOWING = 1, SHADOWING = 3
  -- Si SHADOWING = 2 ou 3, VMS ne change pas la valeur de SHADOWING
  -- Attention au fait que pour supprimer le *shadowing,* il faut que SHADOWING = 0 ET SHADOW_SYS_DISK = 0

Host Based Volume Shadowing

MOUNT :

- Permet la création d'un shadow set, l'addition d'un membre
- Syntaxe identique à celle utilisée pour VMS/HSC *based shadowing*
- Nom de l'unité virtuelle : DSAnnnn
- Nouvelle option : /INCLUDE
  - -- Permet de demander l'inclusion de tous les disques physiques qui étaient précédemment dans le *shadow set*
  - -- Utilise l'information contenue dans la SCB
  - -- Défaut : /NOINCLUDE

- Si le disque de *boot* est un membre valide d'un *shadow set* déjà existant, le noeud qui démarre monte tous les disques qui composent ce *shadow set* automatiquement
- Si le disque de *boot* est un membre valide d'un *shadow set* qui n'a pas encore été créé, le noeud qui démarre crée et monte automatiquement les autres membres du *shadow set* a partir des informations trouvées dans la SCB
- Il n'est ni nécessaire ni recommandé de monter les membres supplémentaires d'un *shadow set* de disque système dans les fichiers de STARTUP

Host Based Volume Shadowing

Migration :

- Pour le moment, le HSC ne procure pas d'assistance au *host based volume shadowing*
- VMS FMR + nouvelle version de CRONIC + ECO protocole MSCP permettront de demander/utiliser l'assistance hardware du controleur
- Il est possible de dissoudre les VMS/HSC *based shadow sets* actuels et de les remonter comme *host based shadow sets* . Nécessite de modifier ses procédures de STARTUP
- Il est aussi possible de garder ses VMS/HSC *based shadow sets* dans leur état actuel

- Certaines configurations nécessitent le *host based volume shadowing* :
  -- Dans le cas d'une configuration dual-CI, les VMS/HSC *based shadow sets* ne sont accessibles que des noeuds CI sur le même CI que le HSC
  -- Par contre, pour la même configuration, les *host based shadow sets* sont accessibles depuis tous les noeuds CI

Host Based Volume Shadowing

Précautions diverses :

- Si l'on passe d'un environnement *shadowing Phase I à shadowing Phase II*, R3 **doit avoir le *byte* de poids fort à 0 et doit pointer sur le numéro d'unité physique**
- Il est recommandé que tous les processeurs d'un cluster bootent sur la même unité physique (simplicité de la gestion du système en cas de *crash*)
- SHADOW_MAX_COPY indique combien d'opérations de type *copy* ou *merge* peuvent etre effectuées simultanément par un VAX : il faut le positionner à une valeur supérieure sur les CPUs les plus puissants et à 0 sur les satellites qui n'ont pas de membres lo-

caux de *shadow sets Phase II*. Un tableau dans le *VMS Shadowing Manual* indique les valeurs conseillées

- Il FAUT modifier le LOAD ASSIST PARAMETER pour pointer sur l'unité virtuelle afin que les satellites puissent booter
- Le process SHADOW_SERVER controle les opérations de *copy* et de *merge*. Ce process tourne sous le compte SYSTEM. Il est recommandé de modifier les parametres UAF comme suit :

BYTLM	SHADOW_MAX_COPY*3KB
BIOLM	SHADOW_MAX_COPY + 10
DIOLM	SHADOW_MAX_COPY + 10
ASTLM	SHADOW_MAX_COPY*2 + 20
TQELM	SHADOW_MAX_COPY*2 + 10
WSQUO	SHADOW_MAX_COPY*128 + 100
WSDEF	SHADOW_MAX_COPY*128/2 + 100
ENQLM	SHADOW_MAX_COPY*2
PGFLQUO	SHADOW_MAX_COPY*130 + 500

- Pour booter un satellite sur un disque système *Phase II* :
  - -- Positionner le *Load Assist Parameter* pour pointer vers l'unité virtuelle
  - -- Positionner SHADOW_SYS_DISK à 1
  - -- Positionner SHADOW_SYS_UNIT en fonction du numéro d'unité virtuelle
  - -- Positionner SHADOWING à 3 pour permettre l'accès aux 2 types de *shadow sets* après le *boot*
- Les membres d'un *shadow set* DOIVENT avoir une *allocation class* NON NULLE
- CECI NE VEUT PAS DIRE LA MEME *allocation class*

Host Based Volume Shadowing

Comment ça marche ? :

La terminologie :

- *Master Member* :
  - -- Membre ayant une SCB garantie correcte
  - -- Membre servant aux "réparations" de données
  - -- Si tous les membres sont **explicite-ment** montes par la même com-mande et que ces membres n'ont ja-mais déjà été membre du *shadow set* c'est le premier périphérique de la liste /SHADOW=()
  - -- Si tous les membres sont **explicite-ment** montes par la même com-

mande et que ces membres étaient déjà membre du *shadow set,* l'ancien *master* reste *master* (idem si le *shadow set* est déjà monté dans le cluster)

-- Si certains des membres sont **implicitement** montes par la commande /INCLUDE et que l'unité virtuelle est montée dans le cluster, le *master* reste le *master*

-- Si certains des membres sont **implicitement** montes par la commande /INCLUDE et que l'unité virtuelle n'est pas montée dans le cluster, le périphérique specifié après le *qualifier* /SHADOW devient le nouveau *master*

-- Pour voir qui est le *master*, utiliser SDA

- *Valid source member* :
  - -- Les données de la SCB sont correctes
  - -- Peut être la source de lectures
- *Merge member* :
  - -- *Valid source member* pendant une operation de *merge;* les opérations de lecture sont traitées spécialement
- Protocole de synchronisation : nécessaire car les I/Os sont distribuées. Utilisé en cas de :
  - -- Besoin pour un VAX d'envoyer une commande MSCP séquentielle en direction de l'unité virtuelle
  - -- Changement d'état du *shadow set*
  - -- Opérations de "réparation" de données
  - -- Transition cluster

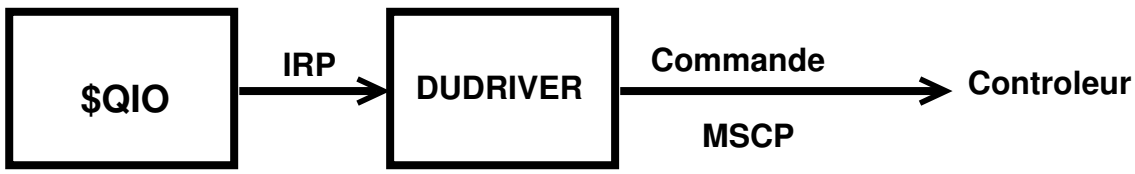
- La synchronisation utilise le *distributed lock manager* à travers une nouvelle interface accessible au niveau *fork*
- L'algorithme est fonctionnellement une commande MSCP séquentielle en direction de l'unité virtuelle :
  - -- Notification des autres VAXes sur lesquels le *shadow set* est monté, du démarrage de la synchronisation
  - -- Les autres VAXes terminent les I/Os en cours et figent les nouvelles I/Os
  - -- Opération séquentielle
  - -- Notification de la fin de l'opération
  - -- Redémarrage des I/Os et validation de l'état du shadow set en cas de changement d'état
- SHDRIVER utilise les structures de données standards (UCBs, VCBs, IRPs,...) pour les membres du *shadow set* et l'unitée virtuelle pour maintenir la com-

patibilité avec DUDRIVER, mais utilise une structure de données : SHAD$

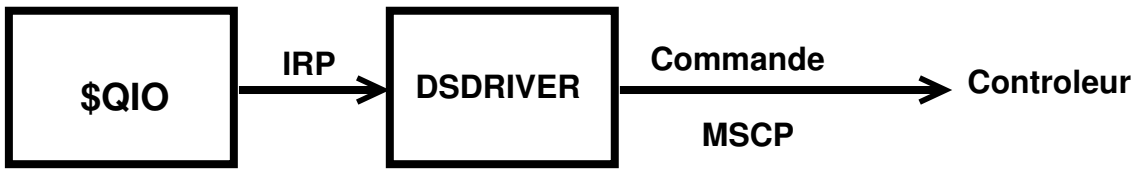
- Une structure de données SHAD$ par *shadow set* connu du VAX et une structure de données SHAD$ par VAX. Cette structure de données est identique sur tous les VAXes sauf pendant les changements d'état du *shadow set*
- Contient :
  - -- UCB/VCB de l'unité virtuelle
  - -- Queue d'IRPs pour l'unité virtuelle
  - -- Informations de *locks*
  - -- "Carte" du cluster
  - -- La SCB :
    - > *Shadow set generation number*
    - > *Unit ID*
    - > Status de l'unité virtuelle
    - > Tableau des membres (UCB, VCB, status,...)

Idée graphique de l'*implémentation* :

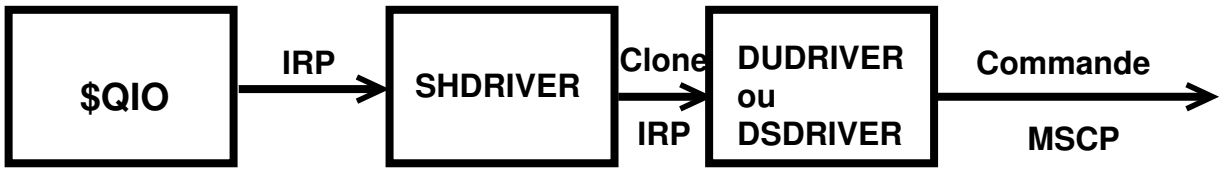
Requêtes d'I/O vers un disque non en *shadow* :



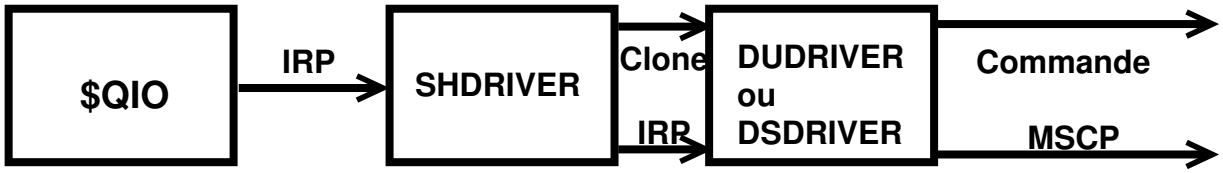
Requêtes d'I/O vers un disque en *shadowing Phase I* :



Requêtes d'I/O de lecture vers un disque en *shadowing Phase II* :



Requêtes d'I/O d'ecriture vers un disque en *shadowing Phase II* :



- Algorithme de lecture :
  - -- Pas de copie en cours :
    - > "Clone" de l'IRP pour lire depuis un *valid source member* ayant la plus petite *queue length* et qui n'a pas été la source de la dernière lecture (SHDRIVER maintient un index du *last read member* dans SHAD$)
    - > Les disques locaux sont favorisés par rapport aux disques *remote*
    - > Algorithme de type *round robin*

-- Opération de *merge* en cours, lecture d'un LBN inférieur ou egal au dernier LBN copié (COPY_LBN dans SHAD$) : algorithme de lecture normal

-- Opération de *merge* en cours, lecture d'un LBN supérieur au dernier LBN copié (COPY_LBN dans SHAD$) :

> Lecture depuis un *valid source member*, commande MSCP *host compare* avec les autres membres du *shadow set*

> En cas de succès, on retourne la donnée a l'utilisateur

> En cas d'échec, on ecrit les données du *master member* sur les *merge members*

- Algorithme d'écriture :
  - -- Pas de copie en cours : on écrit les données sur tous les membres
  - -- Copie en cours :
    - > 2 phases
    - > Ecriture vers les membres qui ne sont pas destinataires d'une *full copy* (Phase I)
    - > Ecriture vers les membres qui sont destinataire d'une *full copy* (Phase II)
- Opération de *full copy* :
  - -- Effectuée sous le controle de SHADOW_SERVER
  - -- Lecture de COPY_LBN LBNs depuis un *valid source member* et stockage en mémoire

- -- MSCP *host compare* avec la même liste de LBNs sur le disque destination
- -- Si *match*, on continue avec les COPY_LBN LBNs suivants
- -- Si *no match*, on ecrit sur le disque destination les LBNs stockés en mémoire et on continue
- -- Dans le cas où la liste de LBNs considérée changerait trop souvent, après 10 itérations, on écrit séquentiellement

- Opérations de *merge* :
  - -- Effectuée sous le controle de SHADOW_SERVER
  - -- Lecture de COPY_LBNs depuis un *valid source member* et stockage en mémoire

-- MSCP *host compare* avec la même liste de LBNs sur le disque destination

-- Si *match*, on continue

-- Si *no match*, on démarre une synchronisation de type *sequential command*

> Relecture depuis le disque source, stockage dans un buffer protégé et ecriture sur les membres

> Fin de la synchronisation *sequential command*

Host Based Volume Shadowing

Licences :

Tout système qui utilise DSDRIVER, SHDRIVER ou les deux doit avoir la license du VAX Volume Shadowing

Host Based Volume Shadowing

Performances en debit :

- Le débit d'I/Os en lecture/ecriture d'un *single member host-based shadow set* est comparable à celui d'un *single member controller-based shadow set*. Les performances dans ce cas sont identiques à celles d'un disque sans *shadow*
- Le debit d'I/Os en lecture d'un *two members host-based shadow set* dans des conditions standards est égal et dans certains cas supérieur à celui d'un *two members controller-based shadow set*
- Le debit d'I/Os en lecture d'un *two members host-based shadow set* alors qu'une *full copy* est en cours est égal et dans certains cas supérieur à celui d'un

two members controller-based shadow set

- Le debit d'I/Os en écriture d'un *two members host-based shadow set* alors qu'une opération de *full copy* est en cours est égal à celui d'un *two members controller-based shadow set*
- Le debit d'I/Os en lecture/écriture d'un *two members host-based shadow set* alors qu'une opération de *full copy* est en cours est nettement inferieur à celui d'un disque sans *shadow*
- Le debit d'I/Os en lecture d'un *two members host-based shadow set* alors qu'une opération de *merge copy* est en cours est en général nettement supérieur à celui d'un *two members controller-based shadow set*
- Le debit d'I/Os en écriture d'un *two members host-based shadow set* alors qu'une opération de *merge copy* est en

cours est égal en général et dans certains cas supérieur à celui d'un *two members controller-based shadow set*

- Le debit d'I/O en lecture/écriture d'un *two members host-based shadow set* alors qu'une opération de *merge copy* est en cours est inférieur à celui d'un disque sans *shadow*
- Dans le meilleur des cas, le temps d'une opération de *full copy* ou *merge copy* pour un *two members host-based shadow set* est 2 fois plus long que le temps correspondant pour un *two members controller-based shadow set*

Mémoire consommée :

- SHDRIVER : 83 pages en *non paged pool*
- SHADOW_SERVER : taille de *working set* maximum :
  PQL_DWSQUOTA+(128*SHADOW_MAX_COPY) = 712 pages par défaut

- Si un ou plusieurs *shadow set* est monté
  - -- Une structure SHAD$ (2 pages)
  - -- Un VCB (240 bytes) par unité virtuelle
  - -- Un UCB (164 bytes) par unité virtuelle
  - -- Un IRP de plus par *outstanding read* (176 bytes)
  - -- Pour un *shadow set* a n membres, n IRPs de plus par *outstanding write*
- Exemple pour un système ayant 6 disques que l'on met en *shadow* :
  - **-- Pas de shadowing :**
    - **> 6 UCBs, VCBs = 5 pages**
    - **> 6 IRPs pour la lecture = 2 pages**
    - **> 6 IRPs pour l'écriture = 2 pages**
    - **> Total : 9 pages**

-- **Shadowing Phase I :**

> **DSDRIVER = 63 pages**

> **18 structures de données (3 par shadow set)= 15 pages**

> **6 IRPs pour la lecture = 2 pages**

> **6 IRPS pour l'écriture = 2 pages**

> **Total : 82 pages, 73 pages de plus**

-- **Shadowing Phase II :**

> **DSDRIVER = 83 pages**

> **SHADOW_SERVER = 150 a 200 pages**

> **18 structures de données (3 par shadow set)= 15 pages**

> **1 SHAD$ par *shadow set* = 12 pages**

> **12 IRPs pour la lecture = 4 pages**

> **18 IRPS pour l'écriture = 6 pages**

> **Total : 270 a 320 pages, 188 a 238 pages de plus que le *shadowing* Phase I**

Consommation CPU :

- Temps CPU moyen en millisecondes pour une opération d'I/O de 8 blocks sur un 8700 ayant 2 RA82s en *shadow* :

Opération	Membres	Pas de shadow	Phase I	Phase II
Lecture	1	0.78	0.76	0.94
	2	------	0.76	0.94
Ecriture	1	0.77	0.75	0.94
	2	------	0.76	1.31

Problemes connus du *host based volume shadowing* :

- Pas de réponse à la commande C sous IPC> pour abandonner un *mount verification*
- Si seulement le disque source devient *write protected* (manuellement ou pour cause d'erreurs) et que la première

opération qui détecte ce problème est une opération de modification de l'état du *shadow set*, le système peut crasher ou passer en *hang*

- Si le numéro d'unité est changé pendant une opération de type *volume processing* (*mount verification*), il y a des risques potentiels de corruption de données car le code ne valide pas correctement la liste des membres du *shadow set*
- Ces problèmes sont corrigés en V5.4-0A

CI load sharing

Buts primaires :

A partir de VMS V5.4, il devient possible d'utiliser plusieurs ports CI par machine. Le *design* permet de supporter 32 ports. VMS V5.4 supporte 4 ports (pour le moment)

Buts :

- Permettre aux ports CI de partager la charge I/O et donc d'éviter les goulots d'étranglement CI
- Permettre de partager la charge equi-tablement entre les differents ports

Buts secondaires :

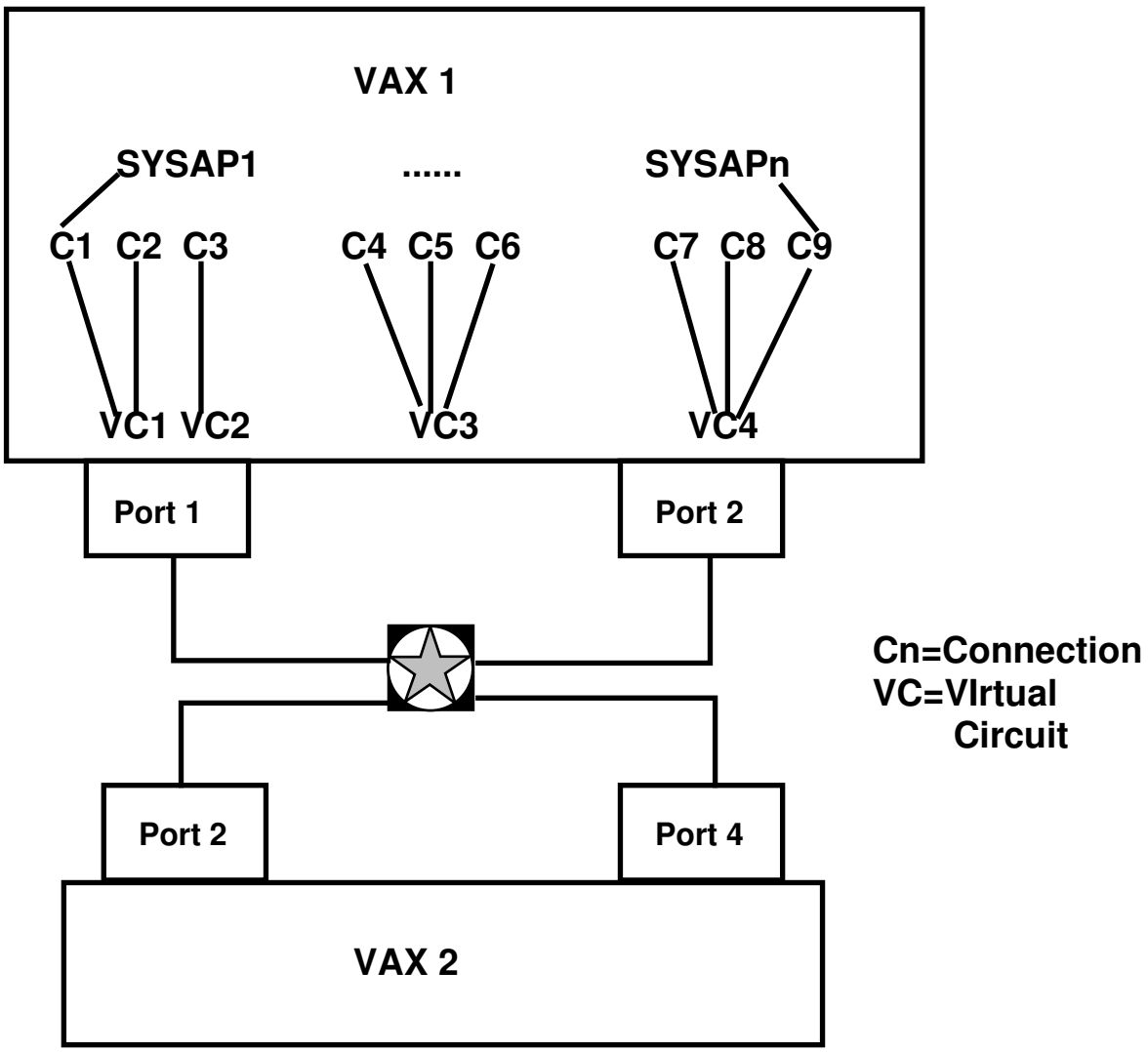
- Améliorer la disponibilité des ports
- Améliorer le temps de reponse I/O moyen

La charge d'un port :

- La charge d'un port CI est affectée par 2 facteurs : saturation et retransmission
  - -- on a saturation quand la charge I/O sur le port dépasse la bande passante du port
  - -- on a retransmission quand un port *remote* est incapable d'accepter un paquet transmis pas un port local
- Le *load sharing* de VMS V5.4 permet de resoudre les problèmes de saturation : les problèmes de retransmission sont invisibles pour VMS

CI load sharing

Rappels SCA :



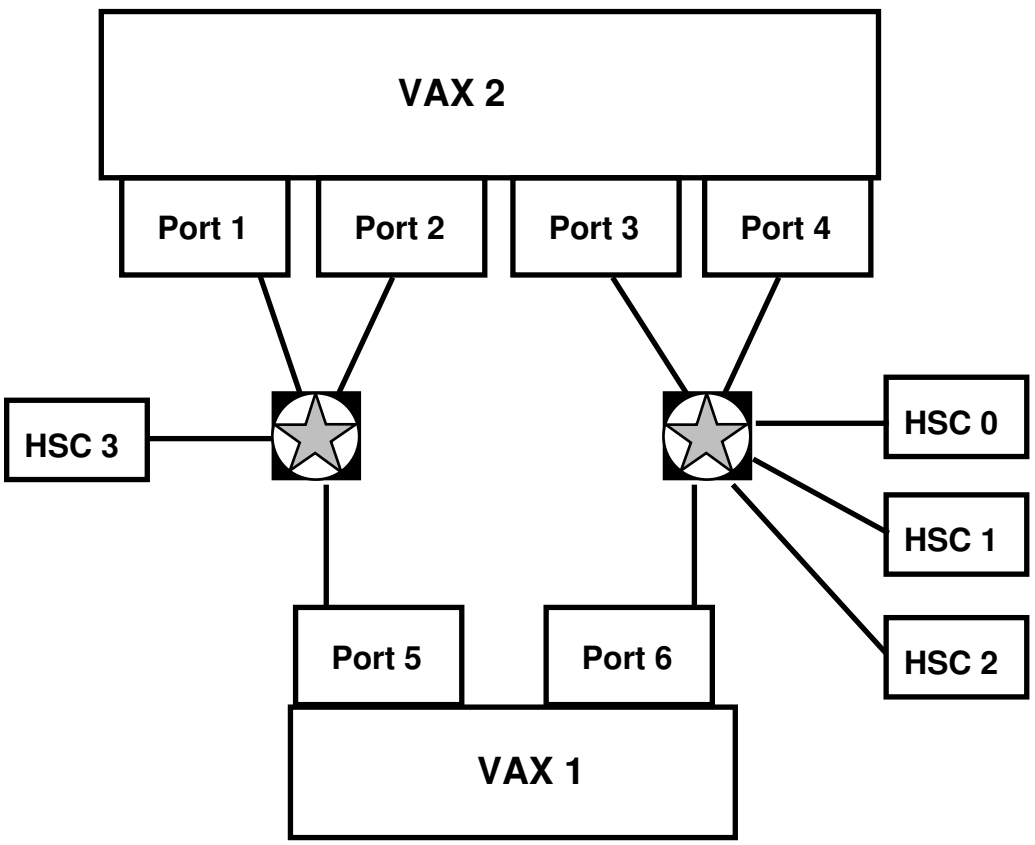
CI load sharing

Notion de *Load Sharing Domain* :

- Un *load sharing domain* consiste en un ensemble de ports qui comprend tous les ports ayant un VC avec au moins un système *remote* commun
- Il peut y avoir plusieurs *load sharing domain* par système
- Le partage de charge est effectué sur chaque *load sharing domain* indépendamment
- Un port SCS peut appartenir a un *load sharing domain* et un seul

CI load sharing

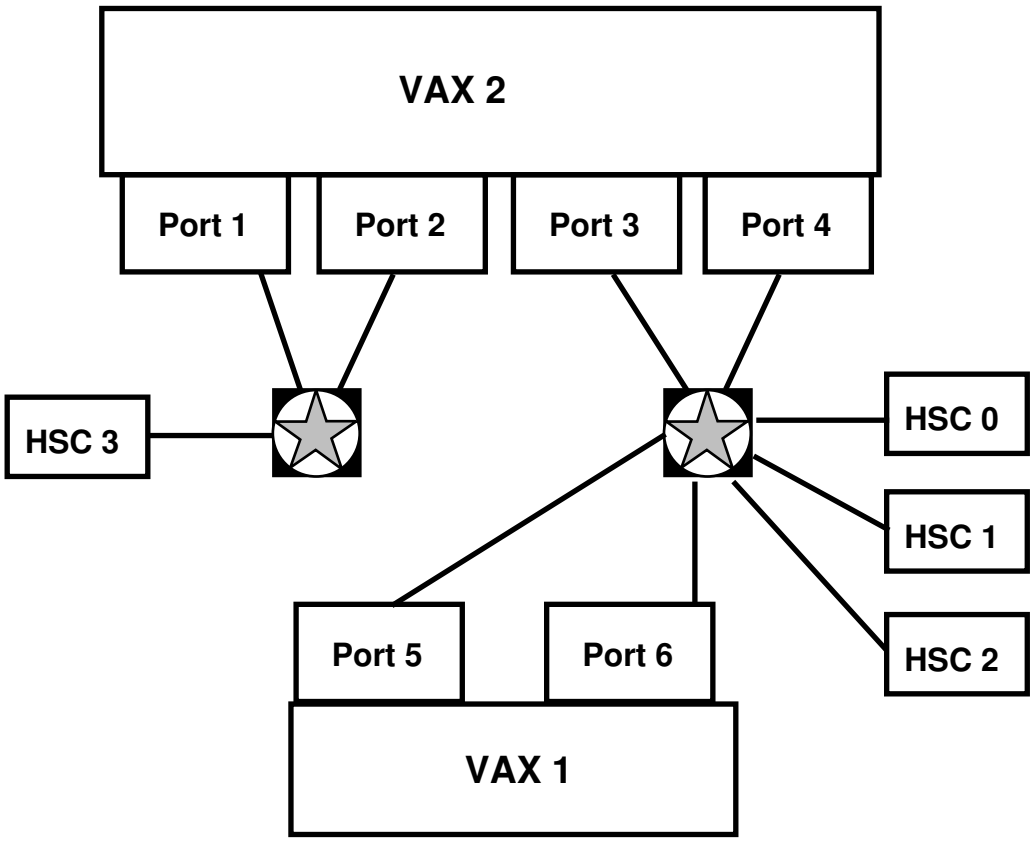
Configuration avec un *Load Sharing Domain* :



(1 et 2) et (3 et 4) ont des VC communs

CI load sharing

Configuration avec deux *Load Sharing Domains :*



CI load sharing

2 Algorithmes utilises :

- Un **Algorithme statique** qui assure que les *connections* sont distribuées parmis les différents ports **avant** la création d'I/Os :
  - -- utilisé lors de l'initialisation du système avant le premier *load sharing interval* (60 secondes)
  - -- utilisé pendant la vie du système lors de la création d'une nouvelle *connection*
- Un **Algorithme dynamique** utilisé tous les *load sharing interval* pour :
  - -- calculer les charges sur les ports
  - -- trouver les *connections* a déplacer

-- démarrer les déplacements de *connections*

- Les déplacements de *connections* sont assurés par la fermeture d'une *connection* et la création d'une nouvelle *connection*
- Aucun parametre SYSGEN (peut etre supprimé en faisant un *clear* du bit 0 de SCS$GL_SHARE_ENABLE)
- Pas de message console
- Pas d'*error logging*

REMARQUE IMPORTANTE : Ces nouvelles fonctionnalités ne sont opérationnelles que si TOUS LES PORTS CI SONT DU MEME TYPE

CI load sharing

Algorithme statique :

- 4 facteurs :
  - -- L'ordre de priorité est CI,DSSI puis NI
  - -- Le nombre de *connections* utilisant déjà chaque controleur CI
  - -- Le type de SYSAP
  - -- Le *connection rating* de chaque controleur CI :

    R = T + k*C

    ou :
    - > T = nombre total de *connections* utilisant un controleur CI
    - > C = nombre total de SYSAPs du même type que le demandeur
    - > k = constante (%x100)

- On utilise le port qui a le plus petit *rating.* Résultat :
  - -- S'il n'y a pas 2 SYSAPs impliquées du même type, les *connections* sont distribuées également parmi les controleurs CI
  - -- Si la nouvelle *connection* est du même type qu'une ou plusieurs *connections* utilisant les controleurs CI, alors on donne priorité à la distribution selon les types de SYSAPs
- Exemple : un VAX ayant 2 ports CI A et B **identiques** sur lesquels il veut creer une nouvelle *connection* MSCP. Le port A a 10 *connections* dont 3 de type MSCP et B a 20 *connections* dont 2 de type MSCP :

  rating sur A : 10 + 256*3 = 778
  rating sur B : 20 + 256*2 = 532

  ==> on forme la *connection* sur B

CI load sharing

Algorithme dynamique : pour chaque *load charing domain* tous les *load sharing interval* :

- On calcule :
  - -- la charge totale du système pour le *domain*
  - -- les caractéristiques de charge du port
- On identifie les ports dans les zones ORANGE ou ROUGE :
  - -- s'il n'y en a pas, l'algorithme est terminé
- Si un port est dans les zones ORANGE ou ROUGE et les autres ports dans la zone NORMALE, on prend la "meilleure connection" pour réduire la charge du port surchargé

- Si tous les ports sont dans les zones ORANGE ou ROUGE :
  - -- on calcule le *port fair share* (partage équitable des ports)
  - -- on prend la "meilleure connection" pour la déplacer de façon à ajuster le partage des ports (pour ceux qui sont défavorisés)
- La "meilleure connection" est choisie a l'aide d'un algorithme spécifique
- Les bandes passantes maximum utilisées pour les calculs de zones sont (pour VMS V5.4) :

Adapteur CI	Saturation (Mbytes/s)
CI780	2.1
CIBCI	1.1
CIBCA-A	1.4
CIBCA-B	2.4
CIXCD	8.0

- La zone NORMALE est définie de 0% a 79%
- La zone ORANGE est définie de 80% a 94%
- La zone ROUGE est définie de 95% a 100%
- De 0 a 79 % les temps de réponses sont de l'ordre de dizaines de ms
- De 80 a 94 % les temps de réponses sont de l'ordre de centaines de ms
- De 95 a 100 % les temps de réponses sont de l'ordre de centaines de ms a quelques secondes
- Le *port fair share* est calculé comme :

$$\text{Point de saturation controleur} * \frac{\text{debit courant total du domain}}{\text{debit maximum du domain}}$$

ce qui revient (si tous les ports sont équivalents) a :

$$\frac{\text{debit courant total du domain}}{\text{\# total d'adapteurs du domain}}$$

- Les compteurs utilisés pour mesurer la charge d'I/O port et *connection* sont :

  > le nombre total de bytes transmis
  > le nombre total de bytes transmis pour DG
  > le nombre total de bytes recus pour DG
  > le nombre total de bytes transmis pour MSG
  > le nombre total de bytes recus pour MSG
  > le nombre total de bytes *mappés* pour BT
  > le nombre total de DGs transmis
  > le nombre total de DGs recus
  > le nombre total de MSGs transmis
  > le nombre total de MSGs recus

- Un *rating* est attribué aux ports :

  -- Zone ORANGE, *Rating* = 4

  -- Zone ROUGE, *Rating* = 6

- Un *rating* est attribué aux *connections* :

  -- *Connection Manager, Rating* = 4

  -- *Disk Class Driver :*

  > Initial, R = 4
  > *Single Member SS*, R = 5
  > *Multiple Member SS*, R= 6
  > *Shadow Copy,* R = 8

- Plus le *rating* est élevé, moins probable est le déplacement de la *connection*
- Un *rating* de 7 permet uniquement un *path balancing* "injuste"
- Un *rating* de 8 supprime complètement le *load sharing* pour la *connection*
- Pendant les déplacements de *connections,* il devient possible pour les disques de devenir *available* pour les controleurs secondaires. Pour éviter que cela ne crée de probleme, DUDRIVER ignore les *attention messages* de ce type pendant les déplacements de *connection*

2 limitations du code VMS V5.4 :

- Si un port devient *online* après que toutes les *connections* aient été créées et qu'aucun des autres ports soient dans les zones orange ou rouge, le code de VMS ne tentera pas de

déplacer les *connections* vers le nouveau port

- Le code du *load sharing* compte les opérations de type *Buffer Map* en se fondant sur l'hypothèse qu'un *Buffer Map* = un seul transfert de type *Block Data*. Dans le cas d'un *shadow set*, cela n'est pas vrai : il y a un transfert *Block Data* par disque physique : cela peut conduire à voir un port moins chargé qu'il ne l'est en réalité

MSCP load balancing

- VMS V5.4 introduit la notion de *MSCP server static load balancing* afin de distribuer plus équitablement la charge due au service MSCP des I/Os entre les différents noeuds servers sans l' intervention du *system manager* avec comme buts de :
  - -- Minimiser le coût quand il n'y a pas d'opération de *load balancing* à faire
  - -- Minimiser l'impact sur le système quand l'opération de *load balancing* est en cours
- Le but du *MSCP server static load balancing* n'est pas :
  - -- d'équilibrer les charges sur les HSCs
  - -- d'équilibrer les charges sur demande
  - -- de donner une interface manuelle

MSCP load balancing

Overview :

- MSCP *server load balancing* permet au *class driver* de choisir le server le moins chargé pendant le choix d'un chemin d'acces (*path*). Appelé *connection walking*
- Démarre lors de l'utilisation de la fonction IO$_PACKACK generée :
  - -- par $MOUNT
  - -- lors d'un *mount verify*
    - > lors d'une erreur I/O
    - > lors d'un *MSCP attention message* indiquant que l'état du disque a changé
    - > lors d'une perte de quorum
    - > ...

- Pendant l'étape de sélection de chemin
  -- on choisit la meilleure (ou la prochaine) *connection* qui satisfait aux critères de base :
     > avoir une *connection* utilisable
     > avoir une *allocation class* d'unité de disque en accord avec celle du serveur (un *host* ne sert que les disques ayant la même *allocation class* que lui)
  -- on envoit une commande MSCP *Get Unit Status* a l'unité pour determiner son status
  -- en cas de succès, on envoie la commande MSCP *Online* et on termine l'IO$_PACKACK
  -- en cas d'echec, on recommence avec l'unité suivante

- Il est possible de modifier le processus de selection de path par des *modifiers* d'IO$_PACKACK :

  -- **LOC_ONLY, ne pas utiliser les chemins MSCP server**

  -- **LOADBAL, utiliser les informations de *load balancing***

  -- **WALK_2P, utiliser le chemin secondaire**

MSCP load balancing

Algorithmes :

- MSCP server maintient un *MSCP server load monitor* :
  - -- *fork process* dans MSCP server
  - -- invoqué par le MSCP server *timer thread*
  - -- calcule la charge I/O du MSCP server
  - -- determine la "valeur de charge disponible" : LOAD_CAPACITY
- La charge est basée sur les requêtes de type *block transfer* (READ, WRITE, COMPARE)
- Chaque I/O peut générer plusieurs *block transfer*

- La "valeur de charge disponible" est calculée à partir de la :
  -- charge d'I/O du MSCP server moyennée sur les 4 derniers *Load Monitor Interval* (20 secondes)
  -- capacité de charge du *MSCP server*
- La capacité de charge du *MSCP server* est determinée de façon statique par la configuration ou par le paramètre SYSGEN MSCP_LOAD
- Cette "valeur de charge disponible" est communiquée aux Class Drivers en réponse aux commandes MSCP périodiques *Get Unit Status*
- Les Class Drivers font un *polling* toutes les 20 secondes pour mettre à jour leur copie locale de "valeur de charge disponible" (dans une nouvelle structure de données : la HULB : *Host Unit Load Block*)

- Pendant la sélection de chemin entre les différents servers disponibles, tous les servers possibles sont comparés et on choisit celui qui a la valeur **la plus grande** de "valeur de charge disponible"
- La copie locale de cette "valeur de charge disponible" est décrémentée quand l'unité passe *online* afin de distribuer les unités entre les différents servers

MSCP load balancing

Parametres SYSGEN affectant le *load balancing* :

- MSCP_LOAD :
  - + 0 : ne pas charger *MSCP server*
  - + 1 : charger *MSCP server* et effectuer le *load balancing* en fonction de la puissance CPU
  - +>1 : charger *MSCP server* et positionner la "valeur de charge disponible" à cette valeur
- MSCP_SERVE_ALL :
  - + 0 : démarrer le *load monitor* quand le server est initialisé
  - + 1 : servir tous les disques et démarrer le *load monitor*
  - + 2 : servir seulement les disques locaux et ne pas démarrer le *load monitor*

- Valeur de LOAD_CAPACITY par défaut

Load_Capacity par défaut	Type de CPU
20	MicroVAX 2000, VS2000
45	11/750, VAXft 3000, MV et VS 3100,VS35xx
60	82xx, 83xx
70	11/780
80	MicroVAX II et VS II
100	11/785, 86xx
130	MV 3300, MV 3400
165	MV 3500, MV 3600, MV 3800, MV 3900, VAX 4000
340	VAX 6000-xxx, VAX 85xx, VAX88xx, VAX 9000

Interface $QIO pour définir un *preferred path* :

- Permet de définir un chemin préféré pour accéder aux disques
- Remplace le *load balancing*
- Uniquement sur le noeud local
- Peut démarrer un *failover* immédiatement
- Peut être utilisée pour faire un équilibrage de charge sur les disques HSC avec quelques restrictions :
  - -- Tous les noeuds sur le même CI que le HSC doivent avoir le même HSC comme chemin préféré
  - -- Cette méthode n'est pas applicable aux unités virtuelles d'un *shadow set* (Phase I ou II)
  - -- Cette méthode n'est pas applicable aux membres d'un *shadow set* Phase I

- Fonction **IO$_SETPRFPATH** :
  - -- Charge le champ UCB$L_PREFCDDB dans l'UCB
  - -- Le *secondary path* sera utilise lors du prochain *failover*
  - -- Un argument : l'adresse du noeud en format *counted string*
  - -- Nécessite le privilège PHY_IO
- *Modifier* **IO$M_FORCEPATH** :
  - -- Démarre le *failover* immédiatement
  - -- Pas d'argument
  - -- Peut impacter les performances du système

- Exemple de code :

```
          .title preferred_path - Set a preferred path
          .enable suppression
          .library  /sys$library:lib/
          .link               /sys$system:sys.stb/
          $dvidef
          $dcdef
          $iodef
          $assigndef
          $qiodef
          $exitdef
;+
;  Macro definitions
;-
.macro    item_def  len,item,buffer,ret; Item definition macro
          .word               len
          .word               item
          .address  buffer
          .long               ret
.endm     item_def


.macro    chk_status          ,?LABEL             ; Do simple error checking
          blbs      r0,label
          $exit_s   r0
LABEL:
.endm     chk_status


.psect    data,long,noexe
;+
; I/O Descriptor definitions
;-

disk_descr:         .word               80        ; LIB$GET_FOREIGN input
                    .byte               14,1
                    .address  disk_buffer
disk_buffer:        .blkb               80


;+
; Text strings
;-

diskprompt_descr:   .ascid    "_Disk : "
not_a_disk:         .ascid    "Specified device is not a disk"
```

```
dvi_list: item_def  4,DVI$_DEVCLASS,device_class,0
                    .long               0

chnl:               .word     0
iosb:               .blkq     1
device_class:       .long 0

node:     .ascic    /PRSSOS/

          .psect    code,long,exe
          .entry    preferred_path,0

          pushaq              disk_descr          ; Get the device name from
          pushaq              diskprompt_descr    ; command line.  If not
          pushaq              disk_descr          ; available, prompt user
          calls               #3,g^lib$get_foreign
          chk_status


          addw3               #disk_buffer-1,disk_descr,r2
          cmpb                (r2)+,#^A/:/
          beql                ok
          movb                #^A/:/,(r2)
          incb                disk_descr

          pushal              disk_descr
          pushal              disk_descr
          calls               #2,g^str$upcase
          chk_status


ok:       $getdvi_s efn       = #1,-              ; Get device data
                              devnam    = disk_descr,-
                              itmlst    = dvi_list,-
                              iosb      = iosb

          $waitfr_s efn       = #1
          cmpl                iosb,#SS$_NORMAL
          beql                CHK_DISK
          movzwl              iosb,r0
          $exit_s             r0

chk_disk:
          cmpl      #DC$_DISK,device_class
          beql      disk_ok
          pushaq    not_a_disk
          calls     #1,g^lib$put_output
          $exit_s
```

```
disk_ok:

          $assign_s devnam=disk_descr,-
                              chan=chnl
          blbc                r0,done
          $qiow_s             chan=chnl,-
                              func=#IO$_SETPRFPATH,-
                              p1=node
          $qiow_s             chan=chnl,-
                              func=#<IO$_SETPRFPATH!IO$M_FORCEPATH>

done:
          $exit_s   r0

          .end      preferred_path
```

Nouveautés Lock Manager

- Changements du *system service* $ENQ
- Changement des algorithmes de *rebuild*
  - -- *Shutdown* et *remastering*
  - -- *Multithreaded rebuild*
- Lock au niveau fork (non documenté)

Nouveautés Lock Manager

Changements de $ENQ :
VMS V5.4 introduit un nouveau flag permettant de controler la façon dont les conversions de locks s'effectuent et ainsi de réduire le coût de l'acquisition d'un nouveau lock

Situation avant VMS V5.4 :

- Une requête de conversion est satisfaite IMMEDIATEMENT si elle est compatible avec les *granted locks*
- Si toutefois un lock actuellement *granted* est converti dans un mode d'accès plus restrictif qui n'est plus compatible avec les *granted locks* existants, le *lock manager* insère ce *converted lock* en FIN de *conversion queue*

- Cela permet à un *granted lock* d'être converti dans un mode compatible ou moins restrictif et d'être *granted* immédiatement sans avoir à passer dans la *conversion queue*
- MAIS, s'il y a beaucoup de *granted locks,* il est possible qu'un *lock* dans la *conversion queue* ne soit jamais *granted* puisque la *granted queue* ne devient jamais vide
- Exemple :
  - -- 3 locks (A,B,C) *granted* en PR
  - -- C converti en EX : incompatible avec PR donc placé en *conversion queue*
  - -- Les routines de *blocking AST* de A et B s'executent car C est bloqué. Dans ces routines, les locks A et B se convertissent en mode NL
  - -- Ces *null locks* sont *granted* car compatibles avec les locks existants dans la *granted queue* (PR) MAIS ne sont

pas *granted* EN MEME TEMPS (le lock manager ne garantit pas cela...)

-- D'abord A est converti, puis B (ou le contraire...)

-- Après que le *null lock* de A ait été *granted,* la *lock blocking routine* de A le convertit à nouveau en mode PR (compatible puisque le seul *granted lock* sur B est en mode *null*)

-- Puis B passe de mode *null* en mode PR (compatible puisque le seul *granted lock* sur A est en mode PR)

-- ET le lock C en attente de conversion en mode EX attend toujours....

Nouveautés Lock Manager

Changements de $ENQ : nouveau flag LCK$M_QUECVT

- Quand utilisé avec LCK$M_CONVERT, specifie que le *converted lock* sera place à la FIN de la *conversion queue* :
  - -- derrière toute conversion en attente
  - -- mis à la fin quelque soit sa compatibilité avec les locks actuellement *granted*
- Avec ce nouveau flag, le programmeur peut :
  - -- s'assurer que les locks dans la *conversion queue* sont traités également
  - -- controler l'ordre des locks dans la *conversion queue*

- Exemple :
  -- 3 locks (A,B,C) *granted* en PR
  -- C converti en EX : incompatible avec PR donc place en *conversion queue*
  -- Les routines de *blocking AST* de A et B s'executent car C est bloque. Dans ces routines, les locks A et B se convertissent en mode NL
  -- Ces locks pour laquel le nouveau flag a ete positionné sont de nouveau convertis en PR MAIS placés derrière le lock C dans la *conversion queue*
  -- La *granted queue* est alors vide et le *lock manager* peut placer le lock C (actuellement dans la *conversion queue*) dans la *granted queue*

Nouveautés Lock Manager

Changements de $ENQ : **nouveau flag LCK$M_EXPEDITE :**
VMS V5.4 introduit un nouveau flag permettant de controler la façon dont les *null locks* sont *granted*

Situation avant V5.4 :

- Un nouveau lock est *granted* si les conditions suivantes sont vraies :
  - -- le nouveau lock est compatible avec les locks existants dans la *granted queue*
  - -- les *conversion queue* et *waiting queues* sont vides

- Ceci peut etre gènant dans le cas ou on demande un *null lock* : puisque le *null lock mode* est toujours compatible avec les autres mode, pourquoi s'en soucier?

Avec VMS V5.4 :

VMS V5.4 indroduit un nouveau flag : LCK$M_EXPEDITE qui indique au lock manager que le *null lock* doit etre immédiatement *granted* :

- la requête de *null lock* ne doit pas être placée dans la *waiting queue*
- quand utilisé avec NLMODE, la requête est immédiatement *granted*
- une fois que le lock est *granted,* il peut etre converti
- ameliore les techniques de conversion de locks

Nouveautés Lock Manager

Shutdown **et** ***remastering*** :

- But :
  - -- Déplacer les ressources *locally mastered* vers un *remote master* pendant le *shutdown* AVANT la transition
  - -- Réduire le temps de reconstruction de la base de données du *distributed lock manager*
- Quand ? :
  - -- Quand OPCCRASH s'execute, un flag CLUB$M_LK_SHUTDOWN est positionné dans la CLUB :
    - > Ce flag n'est pas positionné en cas de *clusterwide shutdown* ou de *bugcheck*

- -- OPCCRASH attend que le *lock manager* ait fini de parcourir les queues des RSBs
- -- Le *lock manager* déplace les ressources *locally mastered*
- -- Le *lock manager* détruit le flag CLUB$M_LK_SHUTDOWN
- -- OPCCRASH continue son suhtdown

- Comment ca marche ?
  - -- On parcourt tous les queues de RSBs
  - -- Le nouveau *master* est selectionné en fonction de la valeur de LOCKDIRWT :
    - > le noeud ayant la plus grande valeur de LOCKDIRWT est choisi
    - > en cas d'egalité, choix aleatoire

-- quand le nouveau *master* est choisi, les operations de *remasterisation* commencent

- Combien de ressources sont *remastered* ?
  -- 50 *remastering threads* à la fois
  -- répété pendant 5 minutes
  -- Les ressources qui sont *remastered* doivent :
    > être *mastered* sur le noeud qui fait le *shutdown*
    > avoir des locks *remote*

Nouveautés Lock Manager

Rebuild :

VMS V5.4 introduit la notion de *multi-threaded rebuild* afin de réduire le temps de transition

Revue des étapes de *rebuild* pendant une transition :

1. Arrêt de toutes les activités de *locking*
2. *Rundown* de toutes les activités de *re-mastering* en cours
3. Démarrage du *rebuild*
4. *Rebuild* des *directory entries*
5. *Rebuild* de la base de donnees du *lock manager* :
   a. *Rebuild* précoce si LOCKDIRWT > 0
   b. *Rebuild* tardif si LOCKDIRWT = 0

6. Redémarrage des activités de *locking*
7. Autorisation des activités de *remastering*

Situation avant V5.4 :

- Le *lock manager* reconstruit les *directory entries* et la base de données du *lock manager* séquentiellement pour chaque ressource (etapes 4 et 5)
- Chaque noeud doit terminer les étapes appropriées sur une ressource avant de passer à la suivante

Avec V5.4 :

- Le *lock manager* devient capable de reconstruire les *directory entries* et la base de données du *lock manager* pour plusieurs ressources en parallèle
  - -- le temps de traitement d'une ressource n'est pas diminué mais plusieurs ressources sont traitées en parallèle

  - -- les données de performances préliminaires indiquent une amélioration de temps de traitement **de 50%**
  - -- les étapes sont encore parcourues séquentiellement
- Le lock manager peut traiter jusqu'a 3 ressources en parallèle. Par exemple, pendant la reconstruction des *directory entries* le *lock manager* demande 3 *directory entries* simultanément
- Quand le *lock manager* effectue ces étapes de reconstruction, il crée une nouvelle structure de données, la CLUPB (Cluster Parallel Block) qui ressemble à un *fork block*
- Quand la *master* CLUPB est crée, le *lock manager* crée 3 *threads* subordonnés (correspondants a 3 CLUPB subordonnés)

- Chacun des threads démarre dans le contexte du *fork dispatcher*
- Comme un *fork* normal, chaque *thread* peut se suspendre ou faire un *re-fork*

Nouveautés sécurité

- Nouveaux *system services* :
  -- $FORMAT_AUDIT
  -- $HASH_PASSWORD
- Filtre de mots de passe
- Divers

Nouveautés sécurité

$FORMAT_AUDIT :

- System service permettant de formatter tous les enregistrements securité du fichier d'AUDIT depuis un format binaire vers un format ASCII avec des possibilité de filtrage des informations trop sensibles (mots de passe par exemple)
- Utilisé par AUDIT_SERVER et ANALYZE/AUDIT
- Implementé sous la forme d'une image partageable VMS$FORMAT_AUDIT_facility ou "facility" désigne une image partageable de formattage
- Pour l'instant une seule "facility" :

  SYS$SHARE:VMS$FORMAT_AUDIT_SYSTEM.EXE

- Plusieurs "facility" peuvent etre actives en meme temps et chargées dynamiquement
- L’image partageable de formattage est constituée de 9 routines permettant l’écriture de "facility" de formattage spécifiques a un site
- Documenté dans le futur avec un exemple (voir $NSADEF pour le format des "packets")

Nouveautés sécurité

$HASH_PASSWORD :

- *System service* utilisé permettant d'appliquer un algorithme de *hashing* sur un mot de passe ASCII et retournant un *quadword* représentant le mot de passe crypté
- Peut être utilisé pour réaliser la certification d'accès à un server ou à un sous-système privilegie
- Appelé par toutes les parties de VMS qui ont besoin d'effectuer la validation du mot de passe (LOGINOUT, AUTHORIZE, NETACP, SYSMAN)
- Support pour des algorithmes spécifiques au site :
  - -- utilisation de $ERAPAT
  - -- exemple dans SYS$EXAMPLES

-- sélection par utilisateur, par mot de passe primaire ou secondaire (sauf le *system password*)

-- sélection de l'algorithme supporté par AUTHORIZE (MODIFY/ALGORITHM) et $SETUAI

Nouveautés sécurité

Filtres de mots de passe :

- Utilisés pour controler la validité de tous les changements de mots de passe lors de :
  - -- $SET PASSWORD
  - -- LOGINOUT (pour changer le mot de passe, pas quand on se connecte)
- Filtres systeme et specifique au site
- Filtre systeme *enable* par defaut
- Pas de filtrage contre les mots de passes auto-generés et contre les mots de passe precisés sous AUTHORIZE ou avec $SETUAI

- **Recherche dans un dictionnaire** :
  - -- tout nouveau mot de passe est verifié contre un dictionnaire de 42979 mots anglais
  - -- fichier sequentiel indexé de 1863 blocks :
    SYS$LIBRARY:VMS$PASSWORD_DICTIONNARY.DATA
  - -- on peut supprimer la recherche dans le dictionnaire par utilisateur à l'aide du flag /FLAG=DISPWDDIC sous AUTHORIZE
- **Recherche historique** :
  - -- Maintien dans le fichier :
    SYS$SYSTEM:VMS$PASSWORD_HISTORY.DATA
    d'un fichier d'historique des derniers mots de passe par utilisateur
  - -- Fichier basé sur le temps plutôt que sur un compte fixe
  - -- Pas de différentiation des mots de passe primaires et secondaires

-- Pas de possibilité documentée de supprimer un enregistrement du fichier historique

-- Par défaut, le systeme maintient la liste des vieux mots de passe pendant 1 an

-- Il y a un nombre maximum de vieux mots de passe que le systeme retient pour un utilisateur donné (pour limiter la place disque consommée)

> Si cette limite est depassée, l'utilisateur est obligé d'utiliser des mots de passe auto-generés
> En V5.4, ce défaut est de 60 mots de passe

-- Possibilité de controle de la durée de vie et la taille de l'historique d'un mot de passe à l'aide de 2 noms logiques systeme (/EXEC)

-

Nom logique	Défaut	Min.	Max.	Unité
SYS$PASSWORD_HISTORY_LIFETIME	**365**	**1**	**28000**	**Jours**
SYS$PASSWORD_HISTORY_LIMIT	**60**	**1**	**2000**	**Nombre**

-- Possibilité de suppression de la recherche dans le fichier historique par utilisateur à l'aide du flag /FLAG=DISPWDHIS sous AUTHORIZE

- Interdiction d'utiliser *username* ou *nodename* comme mot de passe
- Ne pas augmenter la durée de l' historique des mots de passe sans augmenter la taille de l'historique en conséquence :
  -- Duée de vie de mot de passe : 4 ans
  -- Changement de mot de passe : toutes les 2 semaines
  -- Taille limite = 104 pas 60 !!

- Il est aussi possible de valider les mots de passe contre une liste specifique au site. Pour cela, il faut :
  -- charger d'une image spécifique au site :
  SYS$LIBRARY:VMS$PASSWORD_POLICY.EXE

dont le chargement est controlé par un nouveau paramètre SYSGEN : LOAD_PWD_POLICY

-- Ecrire cette image spécifique qui est documentée dans SYS$EXAMPLES (exemple BLISS et ADA)

-- 2 points d'entrées :

> POLICY_PLAINTEXT (filtre les mots de passe en clair)

> POLICY_HASH (filtre la valeur encryptée du mot de passe)

-- Ces routines sont applées par $SET PASSWORD quand un utilisateur change son mot de passe. Elles peuvent donc permettre à l' administrateur securité d'obtenir le mot de passe en clair et son equivalent *quadword*. Il faut donc veiller TRES ATTENTIVEMENT au non remplacement du fichier :

SYS$LIBRARY:VMS$PASSWORD_POLICY.EXE

en placant une ACL par :

```
$SET ACL/ACL=(ALAR=SECURITY,ACCESS=WRITE+CONTROL+-
$_DELETE+SUCCESS+FAILURE) -
$_SYS$LIBRARY:VMS$PASSWORD_POLICY.EXE
```

Nouveautés sécurité

Divers :

- Quand LOGINOUT effectue un changement de mot de passe, il ne demande plus l'ancien mot de passe
- Nouveaux items pour $GETJPI et F$GETJPI :
  >JPI$_RIGHTLIST
  >JPI$_PROCESS_RIGHTS
  >JPI$_SYSTEM_RIGHTS
  >JPI$_LAST_LOGIN_I
  >JPI$_LAST_LOGIN_N
  >JPI$_LOGIN_FAILURES
  >...
- $SHOW PROCESS/RIGHTSLIST
- Les nouveaux mots de passe (reset) sont pré-expirés par défaut

- Modification de l'algorithme de *pre-hashing* : PURDY_V devient PURDY_S
  - -- Le mot de passe est compressé avant d'appliquer l'algorithme de *hashing* (evite d'utiliser du *padding* pour satisfaire les exigences de longueur de mot de passe)
  - -- On fait subir a chaque *longword* du *pre-hash quadword* une rotation à gauche d'un bit après tous les 8 caractères (évite que des mots de passe de longueur supérieure à 8 caractères qui diffèrent de façon symétrique par rapport aux caractères de debut et de fin soient encryptés de la même façon)

Nouveautés RMS

Améliorations caching et locking :

- Les buts :
  - -- Moins de gestion système :
    - > Il est difficile de faire un *tuning* individuel de fichiers
    - > Les mécanismes de *caching* et de *locking* doivent être plus automatiques
    - > Le *tuning* du fichier doit être automatique et suivre l'évolution du fichier
  - -- Meilleures performances :
    - > Optimisation des protocoles de *locking*
    - > Améliorer les mécanismes de bufferisation RMS

Locking :

- RMS utilise intensivement un *lock* : le *file lock*
  - -- nom de la ressource : RMS$+file_id+device_name
  - -- créé par RMS quand un fichier est ouvert de facon partageable ($OPEN ou $CREATE)
  - -- ce *file lock* permet de garder une vue cohérente du fichier à travers son *lock value block* qui contient le marqueur de l'EOF et la longueur de l'enregistrement le plus long
  - -- toujours le parent du *record lock*
  - -- quelque fois le parent du *bucket lock* (quand il n'y a pas de *global buffering*)
  - -- détruit s'il n'y pas d'accédant lors du $CLOSE

- Ce *lock* etait utilisé avant la version V5.4 en mode PW (avec une *blocking AST* associée) ou NL. Le mode PW sérialisait tous les accédants au fichiers même si ces accédants n'avaient pas l'intention de modifier l'EOF
- VMS V5.4 utilise maintenant les modes NL, CR, PR, PW et EX
- CR et PR sont utilisés pour $GET et $FIND (qui ne modifient pas l'EOF)
- PW et EX sont utilisés pour la mise à jour de l'EOF ($PUT, $TRUNCATE)
- $OPEN prend juste un *lock* en mode NL (plutôt que PW !!)
- Ceci permet d'augmenter le parallèlisme des opérations car les *locks* sont compatibles
- Ceci permet aussi de diminuer l'overhead CPU généré (moins de génération de *blocking ASTs*)

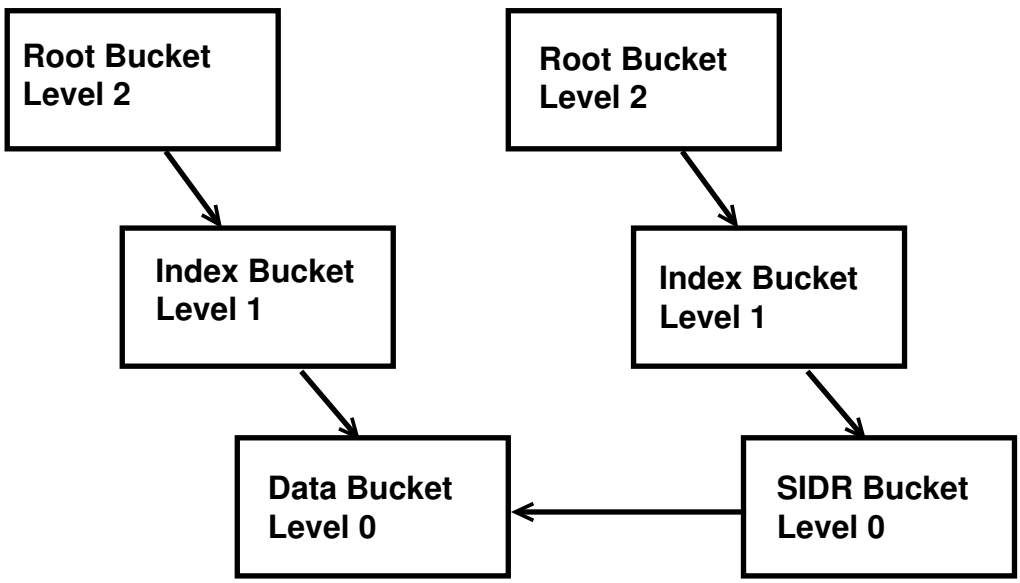
Caching :

- Augmentation du nombre de buffers locaux supportés :
  - -- On passe de 127 a 32767 buffers
  - -- La commande DCL SET RMS_DEFAULT permet un maximum pour /BUFFER_COUNT de :
    - > 127 pour le système
    - > 255 pour un process
  - -- Au niveau programmation, il est possible de préciser : XAB$_MULTIBUFFER_COUNT = 32767 dans $XABITM
  - -- Ceci consomme plus de mémoire : il y a donc un impact potentiel sur *working set quotas* et *pagefile quota*
  - -- Ceci consomme plus de *locks* (chaque buffer local correspond à la création d'un BDB (Buffer Descriptor

Block) et d'un BLB (Buffer Lock Block)

-- Utilisé par Job Control lors d'un : $START/QUEUE/MANAGER/BUFFER=

- Modification du nombre de buffers locaux par défaut pour les fichiers séquentiels indexés :
  -- avant V5.4, le nombre de buffers locaux par défaut etait 2
  -- en V5.4, le nombre de buffers locaux par défaut devient égal au "nombre de niveaux dans l'index le plus profond" + 2



-- Le but est de garder une chaine entière de *bucket* en mémoire

-- Le "+2" sert pour l'accès par cle secondaire plus le *primary data bucket* plus éventuellement un RRV dans le cas où l'on a eu un *bucket split*

-- Un tel algorithme a l'intéret de s'adapter dynamiquement à la croissance du fichier

-- Les tests montrent des reductions du nombre d'I/Os de 35% a 50%

- Modification du buffering pour les buffers globaux :
- Rappel :

-- Le pool de buffer globaux est alloué au $CONNECT sur la base de FAB$W_GBC ou du $SET FILE /GLOBAL=n

-- Ce pool est ordonné suivant l'ordre des VBNs

> La liste des buffers est parcourue séquentiellement pour identifier le buffer ayant le plus

petit "poids" (égal au niveau de profondeur dans la structure)

> La liste des buffers n'est pas ordonnée sous une forme LRU mais les buffers invalides (*deferred write*) ou jamais utilisés sont rassemblés à la fin de la queue. Avant la recherche séquentielle, la fin de la queue est examinée d'abord

> Le type de pool (local ou global) utilisé est basé sur le type de requêtes :

+ les requêtes de lecture utilisent les buffers globaux

+ les requêtes d'écriture utilisent les buffers locaux

- VMS V5.4 introduit une optimisation en utilisant davantage de buffers globaux au niveau des buckets d'index de niveau 1 sur des opérations $PUT et $UPDATE et pour des opérations d'*append* sequentiels quand le mode *deferred write* n'est pas utilisé
- Des tests ont montré des améliorations de performances de 12% a 18%

Pseudo Terminal

Qu'est-ce qu'un pseudo-terminal ?

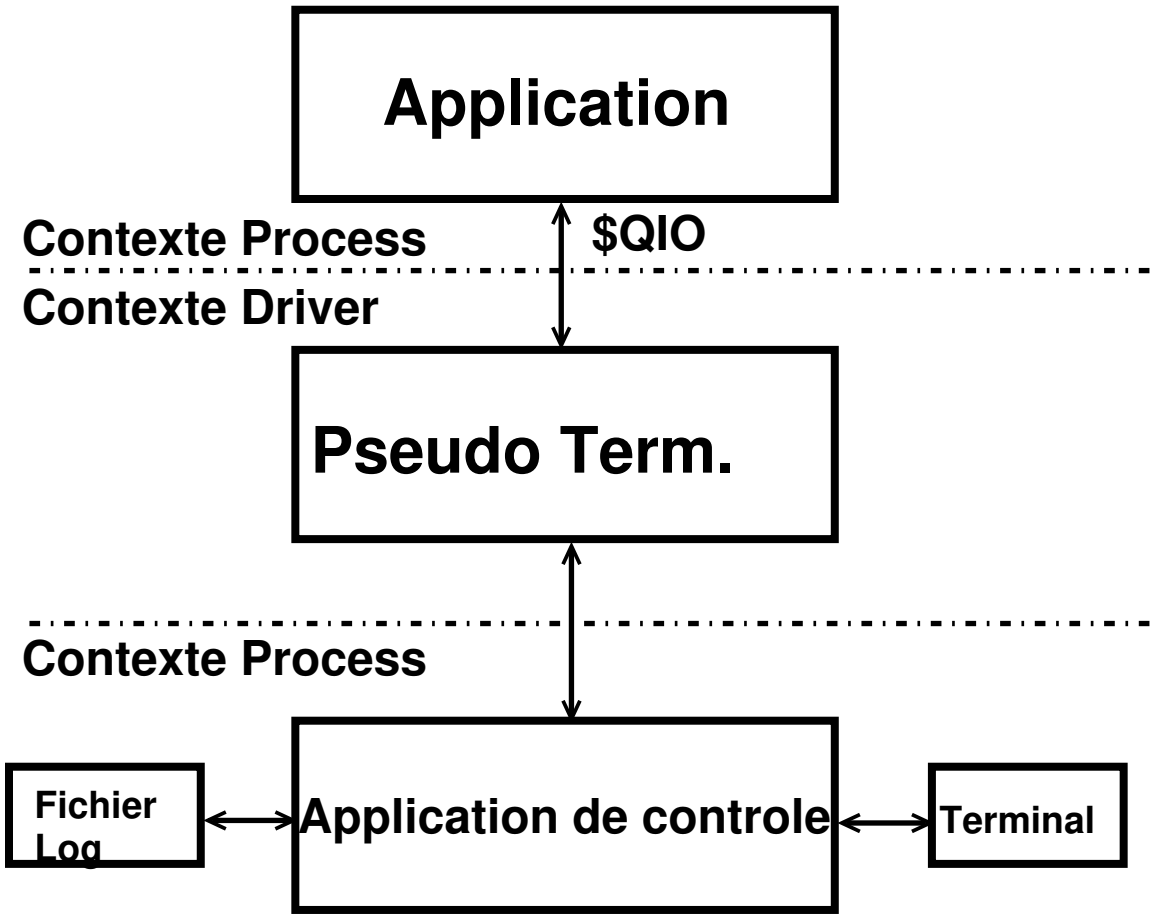
Un périphérique logiciel qui apparait comme un "vrai" terminal pour une application qui communique avec lui mais ne necessite pas la présence du hardware correspondant

Composants :

- Le Pseudo Terminal lui-meme :
  - -- Ensemble de structures de données et de code qui decrivent un terminal
- Une application de controle qui agit de 2 façons :
  - -- Comme un clavier : tout ce qui est écrit en direction du programme de controle apparait sur le pseudo termi-

nal comme si cela avait ete tapé sur un vrai clavier

-- comme un *viewport* vers le pseudo-terminal : tout ce qui est écrit par l'application sera lu par l'application de controle



Les I/Os pour le pseudo terminal sont gerées par l'application de controle :
Enregistrées dans un fichier
Envoyées sur le terminal

Pseudo Terminal

Pourquoi un pseudo-terminal ?

- Pourquoi ne pas utiliser des boites aux lettres :
  - -- Problèmes de performances
  - -- Ce n'est pas un terminal
  - -- Pas de sens de direction (pour le moment)
- Pourquoi ne pas utiliser les pseudo-termiaux déjà existant (DECwindows) ?
  - -- Problèmes de performances (beaucoup de CPU consommé pour renvoyer l'écho des caractères : écriture d'un caractère et écho : 2.7 ms)
  - -- 2 périphériques : PY/TW
  - -- Code dans le domaine public

Pseudo Terminal

Les buts :

Améliorer les performances par un facteur 2 :

- 2 façons d'écrire une application de controle :
  - -- Mode optimal (utilisation des optimisations permettant à 1 *system service dispatch* d'effectuer 2 operations). Ecriture d'un caractère et écho 1.1 ms
  - -- Mode de conversion (conversion de $QIO WRITEBLK en FDT$WRITE et READBLK en FDT$READ). Ecriture d'un caractère et écho 2 ms

Permettre aux applications utilisant le pseudo terminal DECwindows de migrer facilement

Permettre à des applications comme DECterm qui necessitent les privilèges PHY_IO et SHARE pour fonctionner de ne plus en avoir besoin

Ajouter toutes les fonctionnalités possibles

Pseudo Terminal

Les différences :

- Il faut explicitement créer le périphérique
- Il faut utiliser des *system services* spécifiques PTD$ au lieu de $QIO
- On doit utiliser des buffers de taille spécifique (voir plus loin)
- L'*event notification AST* est valide pour la vie du pseudo terminal

Composants :

- Pseudo Terminal Driver (FTDRIVER)
- Librairie partageable de routines SYS$SHARE:PTD$SERVICES_SHR.EXE
- Inclus dans IMAGELIB.OLB
- Démarre automatiquement dans le startup (VMS$DEVICE_STARTUP.COM)

Pseudo Terminal

Les routines :

- PDT$CREATE crée un nouveau pseudo terminal
  - -- nécessite un channel et un buffer aligné sur une frontiere de page pour les buffers d'I/Os
  - -- on peut préciser des caractèristiques de terminal
  - -- on peut spécifier une routine de notification d'AST
- PDT$READ(W) lit les donnéés depuis le pseudo terminal
  - -- nécessite un channel, une adresse de buffer de lecture et une taille

- -- on peut spécifier une routine d'AST de *completion* (couteux, 200 instructions)
- -- on peut aussi spécifier un *efn*

- PDT$WRITE écrit les données vers le pseudo terminal
  - -- nécessite un channel, une adresse de buffer d'écriture et une taille
  - -- on peut specifier une routine d'AST de *completion*
  - -- le mode écho est le moyen le plus efficace d'écrire
  - -- si l'on écrit un byte, il faut etre sur de prévoir un buffer d'echo d'au moins 3 bytes plus long pour permettre les *deletes* (<backspace><delete><backspace>)
- PDT$SET_EVENT_NOTIFICATION autorise ou n'autorise pas les ASTs de *terminal event*

-- nécessite un channel, une adresse de routine d'AST

- PDT$CANCEL supprime les I/Os en queue

  -- nécessite un channel

- PDT$DELETE supprime un pseudo terminal

  -- nécessite un channel

  -- effectue un *marked for delete*

- PDT$DECTERM_SET_PAGE_SIZE change immediatement les longueurs et largeurs de la page

  -- nécessite un channel, une nouvelle longueur et une nouvelle largeur

  -- non documentée

Pseudo Terminal

Considérations de design :

- Controle de flux : ressembler à un vrai terminal
- Nombre de buffers : 6 pages recommandées (1 page en lecture, 1 page en écriture, 4 buffers tournants d'une page pour l'écho)
- Utiliser des ASTs de *completion* pour synchroniser la sortie (c'est la seule façon de garantir des sorties dans l'ordre)

Regles pour les buffers :

- Buffers de 512 bytes alignés sur frontière de page
- Transmission d'un maximum de 508 bytes par requête (premier longword : I/O status : 2 bytes de *byte count* et 2 bytes de *status code*)

Display PostScript

Display Postscript : DPS™:

- Développé et defini par Adobe
- Indépendant du système de fenêtrage
- Présuppose un modèle client-server
- Utilise le meme modèle d'affichage que les imprimantes PostScript©
- Programme en utilisant le même language que les imprimantes PostScript©
- Pour préparer un document à etre visualisé avec DPS :
  - -- "Compiler" les instructions PostScript© avec la commande PSWRAP
  - -- Utiliser la "DPS Client Library" définie par Adobe
- Actuellement utilisé par NEXT™

Display PostScript

XDPS :

XDPS est l'*implementation* par DIGITAL de DPS vue comme une extension de X11 :

- Les parties spécifiques X11 ont ete définies conjointement par DEC/Ultrix, Adobe et DEC/VMS
- Les parties spécifiques X11 ont ete développées indépendamment par Abobe et DEC
- Le protocole XDPS a été soumis au X Consortium par DEC
- Peu de nouveaux operateurs spécifiques a X
- Sous-système de fenêtrage XDPS :

  -- non défini dans DPS lui-même

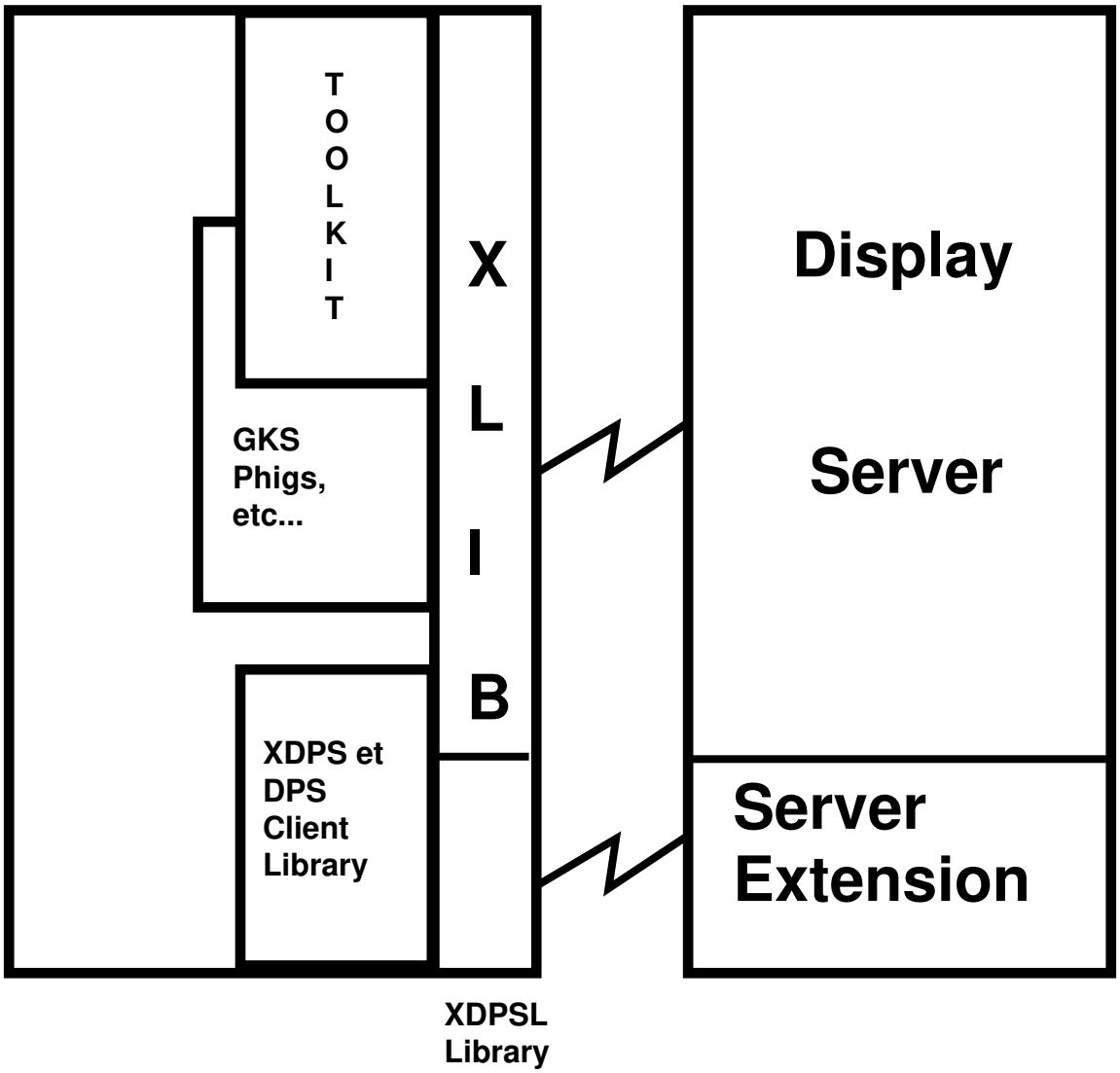
- -- utilise n'importe quelle fenêtre X ou *pixmap* (*drawable*)
- -- on peut mixer des graphiques X et DPS dans un *drawable*

- Capacités graphiques de XDPS :
  - -- opérations simples ou complexes (arcs orientables, courbes,...)
  - -- rotation et changement d'échelle
  - -- *display lists* en definissant un programme PostScript© avec des arguments
- Capacités de gestion de la couleur :
  - -- *true color model* : on spécifie exactement ce que l'on veut
  - -- *dithering* pour obtenir la bonne couleur
  - -- indépendance du moniteur

- Textes (caractéristiques identiques aux imprimantes) :
  - -- beaucoup de *fonts* et styles disponibles
  - -- n'importe quelle taille
  - -- rotation et changement d'échelle a volonté
- Système de coordonnées :
  - -- réelles, définies par l'utilisateur
  - -- basées sur des mesures physiques (indépendant du moniteur)

Display PostScript

L'architecture de XDPS :



Display PostScript

Les ajouts à VMS pour XDPS :

- Display Server Extension
  DECW$SVWT_ADOBE_DPS_EXTENSION.EXE
- Scalable fonts
  SYS$SYSROOT:[SYSFONT.XDPS...]
- Xlib Extension
  SYS$SHARE:XDPS$DPSLIBSHR.EXE
- DPS Client Library (MIT C)
  SYS$SHARE:XDPS$DPSCLIENTSHR.EXE
- DPS Client Library (VAX)
  XDPS$DPSBINDINGSSHR
- Fichier d'*include*
- pswrap (compilateur de code Post-Script©)

Display PostScript

Applications utilisant XDPS :

- CDA Viewer : $VIEW/FORMAT=PS permet de visualiser des fichiers PostScript
- DECwindows Mail : un fichier PostScript commençant par %! peut etre visualisé
- PSDraw : un *exemple* de graphique utilisant XDPS. Sources dans XDPS$EXAMPLES

Un point important a propos de XDPS : ***Comme X11, XDPS est un ensemble*** **de possibilités, pas d'applications !**

Il faut programmer pour en profiter...

Display PostScript

Quelques pièges lors de l'utilisation des applications :

- PSDraw :
  - -- non supporté
  - -- quelques problèmes lors du tracé d'un polygone à beaucoup de faces
- MAIL : %!
  - -- par convention, tous les fichiers PostScript© commencent par !% : sert a identifier les fichier PostScript©
  - -- en cas de *forward* avec *header*, le mail n'est plus reconnu comme PostScript©
  - -- solution : *forward* à soi-meme sans header après avoir selectionné/coupé le *header*

- CDA Viewer et MAil :
  - -- PostScript© contient un sous-language pour définir la structure d'un document
  - -- les imprimantes ne l'utilisent pas en général
  - -- le Viewer l'utilise
  - -- certaines applications le génèrent mal :
    - > pas de marqueur de pages
    - > mauvaise syntaxe sur les marqueurs de pages
    - > pages pas vraiment indépendantes :
      - + les pages ne doivent dépendre QUE du *prolog*

+ les pages doivent laisser l'état de la page comme il l'etait en début (de page)

> problèmes connus :

+ Document V1.1 (V1.2 OK)
+ DECwrite V1.0 (V1.1 OK)

Les performances Consommation memoire

- Pas de changement des besoins minima de mémoire dans le SPD entre V5.3 et V5.4 :
  - -- 2 Mbytes pour VMS Standalone
  - -- +0.5 Mbytes pour DECnet
  - -- +1.5 Mbytes en cluster
  - -- +1.5 Mbytes pour DECwindows si toutes les applications sont *remote*
  - -- +2.0 Mbytes pour DECwindows si certaines applications sont locales
- Accroissement des besoins en mémoire non paginée : 20%, de 1414 pages à 1691 pages, dont 160 pages sont dues à la présence de l'image *loadable* DECdtm

- Paramètres SYSGEN affectant la consommation mémoire :

  -- **valeurs minimum** :

CLISYMTBL	10	==>	20
GBLSECTIONS	20	==>	60
PAGFILCNT	1	==>	4
SRPSIZE	96	==>	112

  -- **valeurs maximum** :

BALSETCNT	1024	==>	8192
IJOBLIM	1024	==>	8192
LOCKIDTBL	262143	==>	262144
LOCKIDTBL_MAX	262143	==>	262144
PHYSICALPAGES	1047552	==>	1048576
VIRTUALPGCNT	600000	==>	1000000
WSMAX	100000	==>	200000

  -- **valeurs par défaut** :

NPAGEDYN	300032	==>	360000
PAGFILCNT	2	==>	4
PHYSICALPAGES	1047552	==>	1048576
SRPSIZE	96	==>	112
VIRTUALPAGECNT	8192	==>	9216

- Nouveaux processes :
  - -- IPCACP et TP_SERVER (DECdtm) sont demarrés avec un *working set* de 2048 pages (1 meg) et ont une taille de *working set* maximum de 4096 pages (2 meg)
  - -- SHADOW_SERVER a une taille de working set maximum de : PQL_DWSQUOTA+128*SHADOW_MAX_COPY Avec les valeurs par défaut, cela fait 712 pages
  - -- Ces processes sont optionnels

Les performances PRIORITY_OFFSET

- En V5.2, le comportement du *scheduler* a changé de telle facon qu'une préemption de process ne s'effectue que si **la différence entre les priorités courantes est supérieure ou égale à 3**
- Par exemple, si le process courant a une priorité courante de 4 et qu'un autre process devient courant à la priorité 6, le nouveau process ne préempte pas le process courant mais attend que le process courant atteigne *quantum end*
- En environnement mixte batch/ interactif, cela peut poser des problèmes
- VMS V5.4 introduit le nouveau paramètre PRIORITY_OFFSET pour permettre l'ajustement de ce comportement

- PRIORITY_OFFSET est positionné à la "différence de priorité courante -1" nécessaire pour faire la préemption
- La valeur par défaut est 0, ce qui signifie le retour au comportement d’avant VMS V5.2

Les performances Impacts du vectoriel

- Consommation mémoire : 16 pages de *nonpaged pool* pour chaque consommateur vectoriel
- Problèmes *scheduling* :
  - -- Un consommateur vectoriel qui est à sa priorité de base (gros consommateur de calcul) a une valeur de *quantum* égale à 32 fois la valeur du paramètre système *quantum* pour :
    - > réduire les besoins en changements de contextes vectoriels
    - > garder le process vectoriel sur un processeur qui est *vector present*
  - -- Un consommateur vectoriel marginal est défini comme un consommateur vectoriel qui a ete VECTOR_MARGIN fois

jusqu'à son *quantum end* sans exécuter d'instruction vectorielle. La valeur par défaut de VECTOR_MARGIN est 100 (pas 200 comme dans le HELP de SYSGEN)

-- Ceci peut poser le problème suivant pour les processes qui n'executent que très rarement des instructions vectorielles :

> un consommateur vectoriel marginal peut consommer beaucoup de temps CPU scalaire sur un processeur qui est *vector present* (3200*QUANTUM ticks) avant de rendre le processeur

-- La seule façon de controler cela est d'utiliser les ACLs sur la *vector capability* d'un processeur qui est *vector present*

- Performances de VVIEF :
  - -- Permet d'émuler les instructions vectorielles sur un processeur n'ayant pas de processeur vectoriel
  - -- Les operations vectorielles émulées par VVIEF correspondent à une vitesse d'exécution égale a 1/5 de la vitesse des instructions scalaires qui effectueraient les opérations similaires
  - -- VVIEF ne doit donc être utilisé qu'à des fins de *debug* : un code qui peut s'exécuter sur des processeurs qui sont aussi bien *vector present* que *vector absent* DOIT tester (via $GETSYI par exemple) l'environnement sur lequel il tourne

Les performances Batch/Print

- De nombreuses améliorations de performances ont été faites depuis VMS V5.2 : seules les améliorations spécifiques V5.4 seront documentées ici
- La taille du *buffer cache* est ajustée dynamiquement. Si l'on ne spécifie pas /BUFFER_COUNT (ou si l'on specifie 0), le Job Controller démarre avec 100 buffers et ajuste ce nombre de buffers en fonction du nombre de queues décrites dans JBCSYSQUE
- Les procédure SYSTARTUP_V5.COM et .TEMPLATE spécifient une valeur de 10 pour /BUFFER_COUNT. Avec les changements V5.4, de nombreux systemes DOIVENT retirer ce *qualifier*

- Amélioration de la gestion du fichier d'*accounting* : *flush* périodique au lieu d'un *flush* pour chaque enregistrement
- Améliorations de performances du *Job Controller* :
  - -- les quotas par défaut du process JOBCTL ont été augmentés
  - -- quand on utilise /DELETE, la suppression des fichiers est repoussée de telle façon que l'on puisse supprimer plusieurs fichiers à la fois et, ainsi, éviter de verrouiller le fichier JBCSYSQUE plusieurs fois

Les performances DECdtm

- Les services de DECdtm (Distributed Transaction Manager) permettent le support d'un protocole 2PC optimisé pour les *ressources managers* DEC. Ces services sont utilisés par RMS Journaling, VAX Rdb/VMS V4.0, VAX DBMS V4.2, DECintact V2.0, VAX RALLY V2.2.
- Les performances de DECdtm sont relativement *linéaires* avec le nombre de CPUS dans un cluster

- Résultats obtenus avec un benchmark (TP1 débit/crédit) utilisant de 1 a 4 processeurs VAX 6000-320 partageant le même disque système, utilisant DECnet comme transport avec 85% des transactions locales et 15% distribuées :

# Noeuds	# Utilisateurs par noeud	%CPU	Debit	%
1	14	96.9	33.7	100
2	12	96.3	60.9	90
3	11	96.0	87.5	87
4	11	96.3	113.2	84

Les performances MSCP

- Réduction du nombre de messages : avant VMS V5.4, dans une opération de type READ, MSCP server envoyait un *acknowledge* séparé apres le dernier block transmis. Avec VMS V5.4, ce message est envoyé (*piggyback*) avec le dernier block
- Comme la plupart des commandes MSCP de type READ sont effectués avec des transmissions d'1 block, il y a une diminution substantielle du nombre de messages
- Cette modification réduit la charge CPU et améliore le débit des commandes MSCP READ de 9%

- Consommation mémoire pour le service des disques MSCP (identique a V5.3) :
  - -- 23 pages pour MSCP server
  - -- 2/3 pages pour chaque *connection*
  - -- 1/4 pages pour chaque unité servie
  - -- 26 bytes pour chaque paire *host/unit*
  - -- 1/2 page pour chaque *request packet*
  - -- 128 pages (par défaut) pour le pool de buffers

Les performances VAX 9000

Espace mémoire :

- Le VAX 9000 est capable de supporter 512 Mbytes de mémoire :
  - -- plus de mémoire = plus grand nombre d'utilisateurs possibles. Ceci implique une plus grande taille des structures de données telles que PCB *vector*, *balance set*
  - -- plus de mémoire = plus grande taille de *working sets* possible. Ceci implique une plus grande taille des structures de données telles que *process headers* et *global page table*
  - -- tout cela consomme de l'espace S0
  - -- Les valeurs initiales d'AUTOGEN ne sont pas assez grande : il peut être

nécessaire d'utiliser plusieurs itérations d'AUTOGEN-feedback

Vitesse du CPU :

- La vitesse du CPU fait qu'il est possible de faire beaucoup de choses en 10 millisecondes : ceci peut conduire, si l'on utilise les défauts, à un système suboptimal

Quelques recommendations

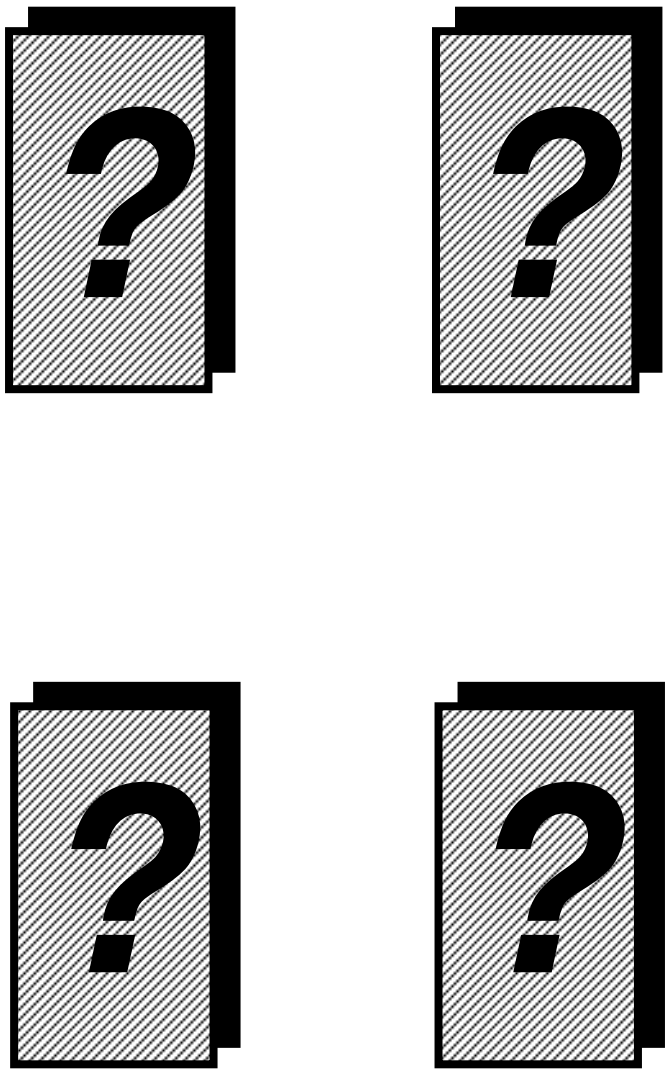
- Voici quelques idées de paramètres initiaux pour un VAX 9000 :

Parametre initial	Valeur
MIN_SRPCOUNT	3000
MIN_IRPCOUNT	1500
MIN_LRPCONUNT	100
MIN_MAXPROCESSCNT	400
MIN_NPAGEDYN	5000000
MAX_VIRTUALPAGECNT	300000
MIN_BALSETCNT	350
MIN_SYSMWCNT	2000
MIN_SCSCONNCNT	40
MIN_LNMSHASHTBL	2048

QUANTUM	5
MIN_ACP_HDRCACHE	1500
MIN_ACP_DIRCACHE	1500
MIN_ACP_DINDXCACHE	300
MIN_ACP_MAPCACHE	300

- Ces valeurs permettent de favoriser un environnement comprenant un grand nombre de processes avec un espace virtuel par process limité (VIRTUALPAGCNT limité)
- Si l'on désire supporter de grosses applications nécessitant un plus gros *Working set*, on peut augmenter VIRTUALPAGCNT tout en diminuant BALSETCNT

Conclusion

- Voici un aperçu des nouvelles fonctionnalités de VMS V5.4
- Bientôt, une nouvelle version encore plus importante... BFMR (Big FMR...)
- Rendez-vous au symposium pour en parler... et pour la voir...tourner...

Questions