DECram

Overview

Bernard OURGHANLIAN

DECUS SIG VAX

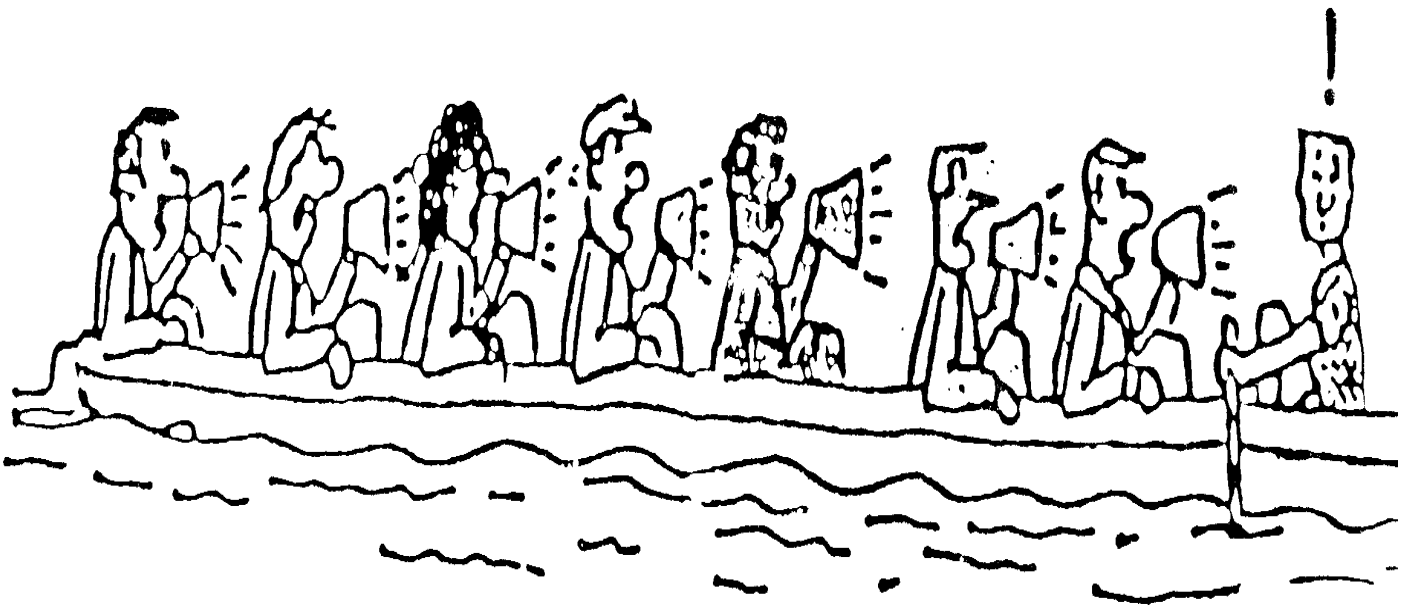
15 Octobre 1991

Sommaire

1. Introduction
2. Pourquoi DECram ?
3. Le paramètrage et la configuration d'un disque DECram
4. L'utilisation d'un disque DECram
5. Gestion système autour de DECram
6. Comment çà marche ?
7. Performances

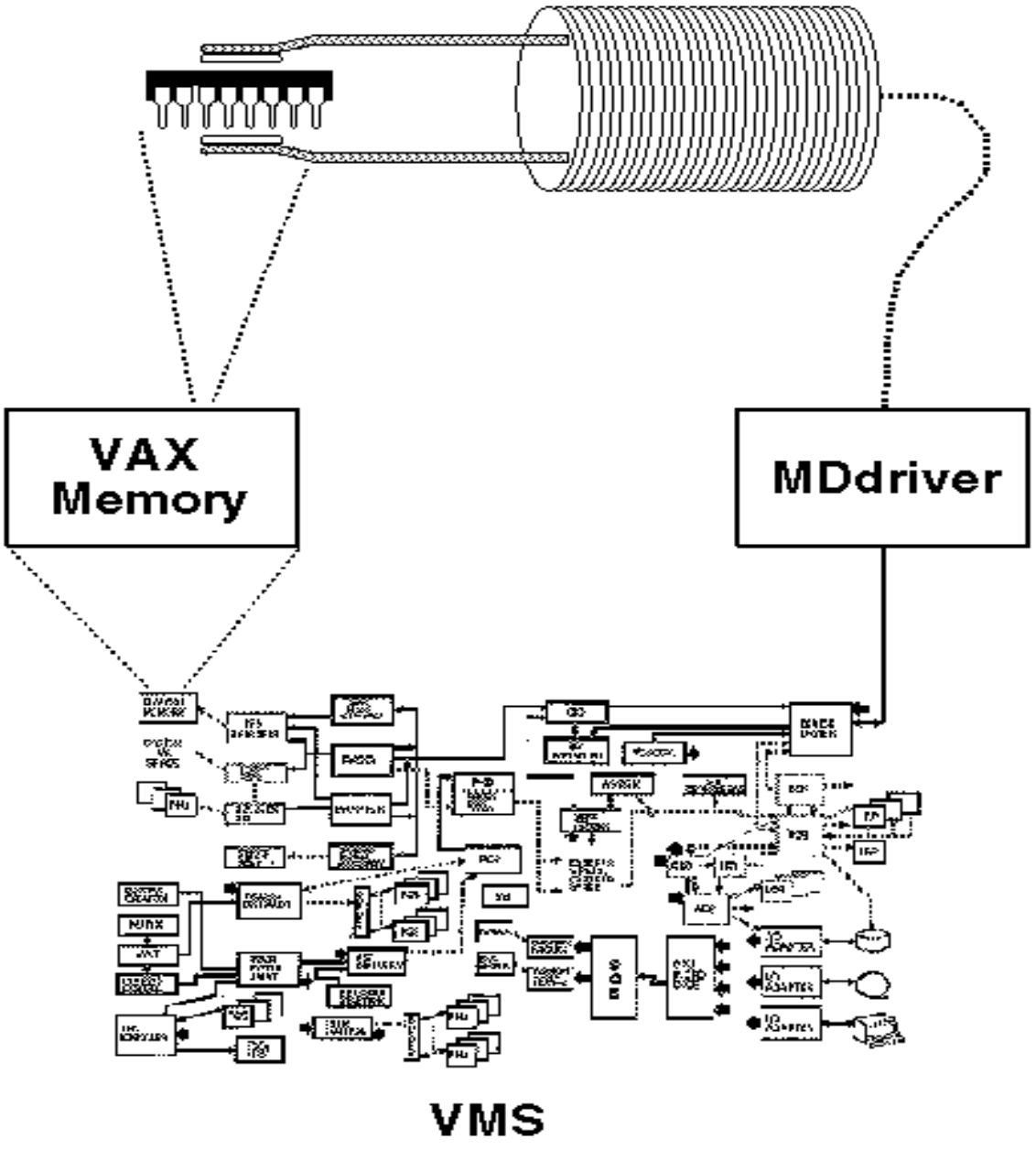
Introduction

DECram n’est pas cela.....



Introduction

DECram est quelque chose comme cela.....



Introduction

- DECram est essentiellement un *driver* (MDDRIVER) permettant de créer un disque en **mémoire**
- Un disque DECram peut être utilisé à travers le *file system* VMS de la même façon que les disques physiques **sans aucun changement applicatif**
- Comme les données d'un disque DECram résident en mémoire centrale, il est clair que ces données seront **perdues en cas de *powerfail* ou de *crash* système** : il ne faut donc stocker sur ce type de disque que des fichiers *read-only* ou des fichiers temporaires (*scratch*)

Pourquoi DECram ?

Quelques données technologiques :

Technologie	Densité et performances
Technologie d'intégration de circuits	**Augmentation du nombre de transistors sur 1 chip de 25%/an. Doublement tous les 3 ans**
Technologie de semi-conducteurs DRAM	**Augmentation de la densité de presque 60%/an. Quadruplement tous les 3 ans. Amélioration du temps de cycle très lente : 1/3 en 10 ans**
Technologie des disques	**Augmentation de la densité de 25%/an. Doublement tous les 3 ans. Amélioration du temps d'accès très lente : 1/3 en 10 ans**

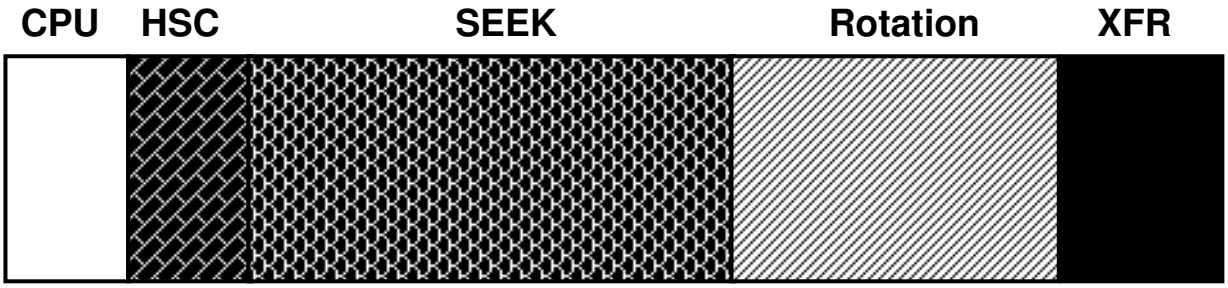
- La puissance des CPUs augmente rapidement
- La capacité de la mémoire augmente rapidement
- La capacité disque augmente rapidement

MAIS...

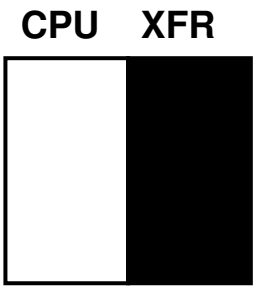
Le temps d'accès aux disques n'augmente pas dans la même proportion

Pourquoi DECram ?

Les performances :



Disque magnétique



DECram

Pourquoi DECram ?

	DECram	Disque magnétique
Type de périphérique	**Disque virtuel qui fonctionne comme un disque magnétique**	**Disque magnétique classique**
Paramètrage du disque	**A chaque démarrage du système, on doit charger les fichiers**	**Lors de l'installation**
Modèles disponibles	**Un seul modèle disponible; on peut choisir la taille**	**Plusieurs modèles disponibles chacun étant de taille fixe**
Utilisation comme moyen de stockage	**Contient typiquement des petits fichiers; extension limitée**	**Contient des fichiers de toute taille**
Opérations d'écriture	**Le stockage des données est volatile; Pas de fichiers permanents**	**Stockage permanent**
Performances	**Extrèmement rapide; limitées par la puissance CPU et la bande passante mémoire**	**Vitesse limitée par des considérations mécaniques : temps de déplacement et de rotation**

Pourquoi DECram ?

	DECram	Cache Disque
Type de périphérique	**Disque virtuel qui fonctionne comme un disque magnétique**	**Fonctionne comme la mémoire**
Paramètrage du disque	**A chaque démarrage du système, on doit charger les fichiers**	**Facile à configurer; cache transparent aux utilisateurs**
Types de données	**Possibilite de choix des fichiers; contrainte de taille**	**Pas de choix de fichier possible en général**
Utilisation comme moyen de stockage	**Contient typiquement des fichiers entiers; taille limitée; contient les *blocks* de données de fichier et les *blocks* du *file system***	**Contient typiquement des parties de fichiers; contient en général les *blocks* de données et pas les *blocks* du *file system***
Opérations d'écriture	**Le stockage des données est volatile; Pas de fichiers permanents**	**Stockage permanent**
Performances	**Toujours prédictible**	**Non toujours prédictible; dépend de la taille du cache**

Paramètrage d'un disque DECram

Préparation

Déterminer la taille du disque à créer :

- Tout *block* disque alloué au disque DECram consomme une page supplémentaire de la mémoire du VAX
  - — Il est possible de créer un disque DECram de taille maximum 524288 *blocks* (256 Mbytes)

Déterminer la consommation en ressources système :

- DECram est un *driver*, c'est à dire un ensemble de *routines* et de tables utilisées par VMS pour traiter les demandes d'I/Os pour un disque DECram
- En plus des besoins en *nonpaged pool* pour le chargement du code du *driver* et pour

l'ensemble des structures de données de l'*I/O database*, un disque DECram consomme une page supplémentaire de la mémoire du VAX par *block* disque :

— Cette mémoire est prise depuis la *free list*, pas depuis la *nonpaged pool*

— Ceci permet de "rendre" la mémoire prise par un disque DECram si on n'en a plus besoin (voir plus loin)

- Un disque DECram a besoin de SPTEs (*System Page Table Entries*) supplémentaires :

— Ce nombre de SPTEs supplémentaires est lié à la taille du disque DECram que l'on désire créer :

> 1 SPTE par 128 pages mémoire allouées au disque DECram +

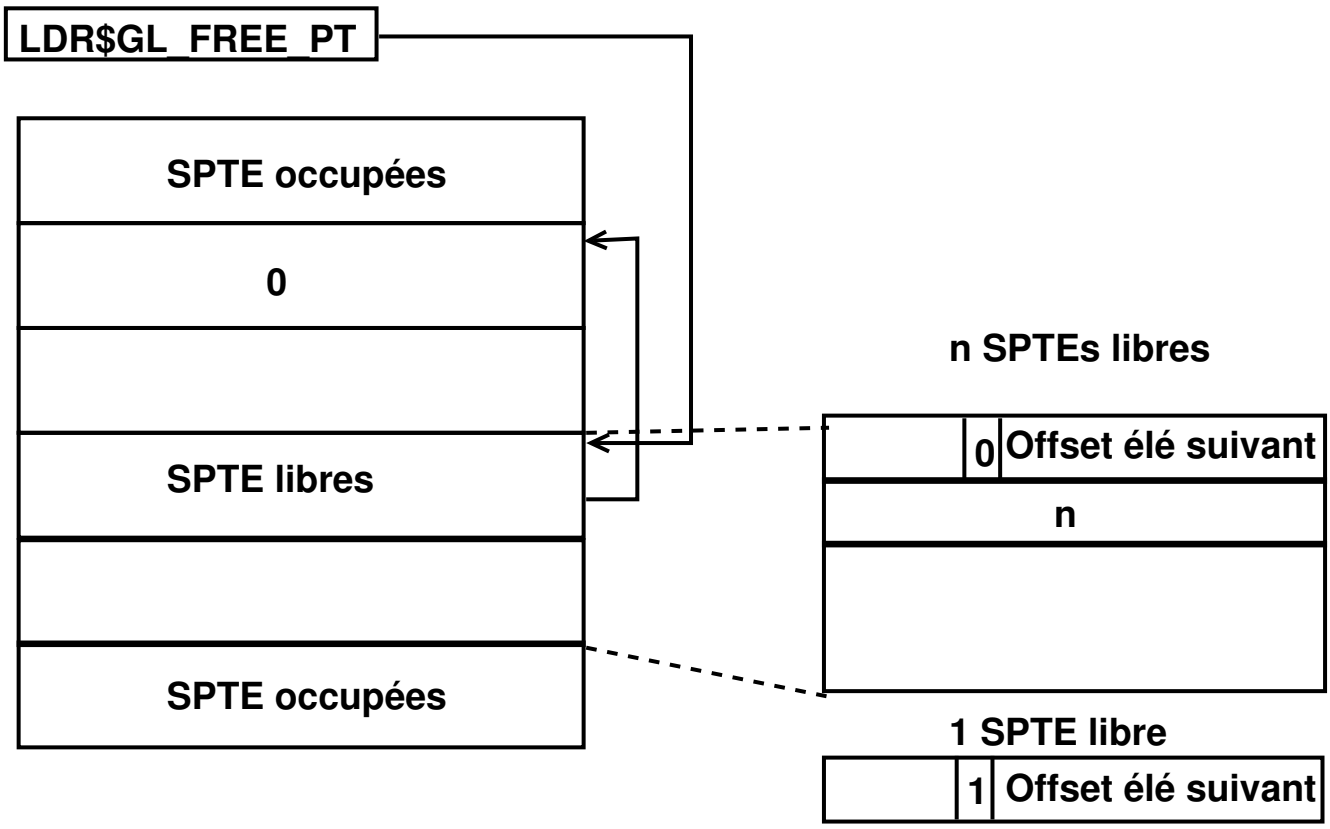
> 128 SPTEs pour *mapper* les *buffers* utilisateurs pendant le transfert de données

— Ces SPTEs doivent être **contigües**

- *Quelques Rappels :*
  - *Depuis VMS V5.0, il est possible d'allouer et de désallouer dynamiquement des SPTEs : ceci permet de disposer de SPTs que l'on peut allouer pour charger une "loadable executive image" et désallouer sa routine d'initialisation et sa section de "fixup" quand celles-ci ne sont plus nécessaires*
  - *La liste des pages d'espace système disponibles est maintenue à l'intérieur même des SPTEs disponibles*
  - *2 routines permettent de gérer la liste de pages disponibles :*
    - *LDR$ALLOC_PT qui alloue les SPTEs*
    - *LDR$DEALLOC_PR qui désalloue les SPTEs*
  - *LDR$GL_FREE_PT est le "longword" qui pointe sur le premier élément de la liste des SPTEs libres :*

> *Un élément de cette liste est en général composé de 2 "longwords" : le premier pointe vers le prochain groupe de SPTEs libres et le second indique le nombre de SPTEs dans le groupe considéré; dans ce cas, le bit 21 de la SPTE est à 0*

> *Quand le groupe de SPTEs libre ne contient qu'un seul élément, le bit 21 de la SPTE est à 1*



- *La routine suivante permet d'obtenir le nombre de SPTEs libre de son système :*

```
          .SBTTL    Definitions

          $IPLDEF
          $LDRPTEDEF
          $SSDEF

          .SBTTL    $$DATA    RD,WRT,NOEXE

KERNAL_ERROR:       .ASCID    /Kernal mode code error, process will be vaporized./
KERNAL_ERROR_2:     .ASCID    /Hopefully, just your process went and not the system./
FAOCTL:             .ASCID    /Croaked at: !XL, Accessing Loc: !XL/
ULOGY:              .ASCID    /                                                   /
DEADPC:             .LONG     0
BADADD:             .LONG     0

LOCK_ADDR:
          .ADDRESS  BEGIN_KERNEL_CODE
          .ADDRESS  END_KERNEL_CODE

          .PAGE
          .PSECT KERNEL_HANDLER
          .SBTTL Kernel mode exception handler
          .ENTRY HANDLER, ^M<>
;+
;         Prints out the offending PC and memory location causing the exception.
;         The only one I'm interested in is access violations.  If the exception
;         was NOT an accvio, I'll just die on the spot.  First drop IPL to 0
;         (it cant hurt if already there), just in case I'm at elevated IPL.
;-
          MTPR      #0, #18                             ; lower IPL just in case.
          PUSHAQ    KERNEL_ERROR                        ; Embarrass user right
                                                        ; away.
          CALLS     #1, G^LIB$PUT_OUTPUT
          MOVL      4(AP), R11                          ; Get address of signal
                                                        ; array.
          CMPL      4(R11), #SS$_ACCVIO                 ; Trouble if not ACCVIO
          BEQL      1000$                               ; Branch if IS ACCVIO
          $EXIT_S                                       ; Otherwise die (exit
                                                        ; VMS).
1000$:    MOVL      12(R11), BADADD                     ; Get offending address
          MOVL      16(R11), DEADPC                     ; and Program Counter
          $FAO_S    CTRSTR = FAOCTL, -                  ; Format a pretty message
                    OUTBUF = ULOGY,  -                  ; for the user containing
                    P1     = DEADPC, -                  ; offending PC ...and...
```

```
                P2      = BADADD                        ; offending address
        PUSHAQ  ULOGY                                   ; print the pretty message
        CALLS   #1, G^LIB$PUT_OUTPUT                    ; on the screen
        PUSHAQ  KERNEL_ERROR_2                          ; this other message is
                                                        ; just
        CALLS   #1, G^LIB$PUT_OUTPUT                    ; so the terminal server
                                                        ; doesn't
                                                        ; take over too quick and
                                                        ; suppress the PC/Address
                                                        ; message.
        $EXIT_S

        .SBTTL  GET_FREE_SPT -- Main Routine. (Go into kernel mode)

        .PSECT  $$CODE    PIC,NOWRT,RD,EXE
        .ENTRY  GET_FREE_SPT,^M<>
;+
;  Lock down the KERNEL code
;-
        PUSHR   #^M<R2,R6>                              ; Save registers.
        $LCKPAG_S                       -               ; Lock KERNEL mode routine
                INADR=LOCK_ADDR                         ;
        MOVAL   4(AP), R2                               ; Get parameter address.
        $CMKRNL_S B^BEGIN                               ; Go into kernel mode
        CMPL    #SS$_NORMAL,R0                          ; Is it normal complete ?
        BEQL    5$                                      ; Yes
        CLRL    R6                                      ; Nope, return with count
                                                        ; = 0.
5$:
;+
;   Unlock pages
;-
        $ULKPAG_S                       -               ; Unlock KERNEL code
                INADR=LOCK_ADDR                         ; routine
        MOVL    R6, @(R2)                               ; Pass it back to caller.
        POPR    #^M<R2,R6>                              ; Restore registers.
        RET

        .SBTTL  BEGIN -- $CMKRNL invokes this routine to get free SPTE's.

BEGIN_KERNEL_CODE::
        .ENTRY  BEGIN,0
        MOVAL   HANDLER, (FP)                           ; Exception handler.

        PUSHR   #^M<R2,R3,R4,R5>                        ; Save some registers.
        LOCK    LOCKNAME=MMG,-                          ; Synchronize with all
                                                        ; CPUs.
                LOCKIPL=IPL$_SYNCH,-                    ; Raise it to SYNC.
                SAVIPL=-(SP)                            ; Save current IPL.
        MOVAB   G^LDR$GL_FREE_PT, R3                    ; The first free SPTE.
```

```
        TSTL    (R3)                                    ; Empty ?
        BNEQ    10$                                     ; Nope, There are more.
        MOVL    #SS$_INSFSPTS, R0                       ; No entry status.
        BRB     60$                                     ; On my way home.
10$:
        ASHL    #2, (R3), R4                            ; For VMS V5.4
        ADDL    G^LDR$GL_SPTBASE, R4                    ;
        CLRL    R5                                      ; Initialize free SPTE
                                                        ; count.
        CLRL    R6                                      ; Init. MAX free SPTE
                                                        ; count.
20$:
        BBS     #LDRPTE$V_FLAG, (R4), 30$               ; Is it single spte ?
        MOVL    LDRPTE$L_COUNT(R4),R5                   ; Nope, get count.
25$:
        CMPL    R5, R6                                  ; Compare counts.
        BLEQ    40$
        MOVL    R5, R6                                  ; Save MAX count.
        BRB     40$
30$:
        MOVL    #1, R5                                  ; Get count.
        BRB     25$
40$:
        EXTZV   #LDRPTE$V_LINK, #LDRPTE$S_LINK,-        ; Get link to next free
                (R4), R4                                ; one.
        BEQL    50$                                     ; End of list.
        ASHL    #2, R4 , R4                             ; For VMS V5.4
        ADDL    G^LDR$GL_SPTBASE, R4                    ; Again, address of free
                                                        ; SPT.
        BRB     20$                                     ; Loop.
50$:
        MOVL    #SS$_NORMAL, R0                         ; Signal success.
60$:
        UNLOCK  LOCKNAME=MMG,-                          ; Desynchronize system
                                                        ; structures
                NEWIPL=(SP)+,-                          ; Restore our IPL.
                CONDITION=RESTORE                       ; Conditionally release
                                                        ; spinlock
        POPR    #^M<R2,R3,R4,R5>                        ; Restore registers.
        RET

END_KERNEL_CODE::

        .END
```

- *Cette routine peut être appelée par le simple exemple suivant :*

```
Istatus = Get_Free_Spt(I_Spt)
Type *,'# of Free SPTEs = ', I_Spt
End
```

- Les besoins en SPTEs pour un disque DECram sont donnés par la formule suivante :

  128+((Taille du disque DECram en blocks +127) /128)

- Exemple :
  - Si l'on veut créer un disque DECram de 10000 blocks (5Mbytes) :

  SPTEs = 128 + ((10000+127) / 128) = 207

  Il faut donc augmenter SPTREQ de 207

  - Si la valeur courante de SPTREQ ne permet pas de disposer de suffisamment de SPTEs (voir programme donné plus haut), il faut augmenter SPTREQ en conséquence, ce qui nécessite un *reboot* du système
  - Compte tenu de la nature du format des SPTEs, il est recommandé de positionner SPTREQ comme un multiple de 128 : le

coût de la mémoire "perdue" est négligeable

— Ici, il faut donc éditer le fichier MODPARAMS.DAT et ajouter la ligne:

ADD_SPTREQ=256

et invoquer AUTOGEN

- DECram utilise, comme on le verra plus loin des *private page table entries*, ce qui évite d'utiliser une SPTE par *block* du disque

Création d'un disque DECram :

- Pour créer un disque DECram, utiliser la commande SYSGEN

  CONNECT/NOADAPTER

  SYSGEN> CONNECT MDcuu/NOADAPTER
  où :
  c : Toute lettre de A à Z
  uu : 2 digits

Allocation et désallocation d'espace pour un disque DECram :

- Un disque DECram "consomme" une page mémoire par *block* d'espace disque alloué
- Pour allouer l'espace pour un disque DECram, utiliser la commande :

  $INIT/SIZE=n device-name label

  où n désigne la quantité de mémoire allouée au périphérique

- n doit être inférieur à 524288 : si l'on désire utiliser un disque DECram de taille supérieure, il faut utiliser un *volume set* de plusieurs unités DECram
- Pour désallouer l'espace mémoire utilisé pour le disque DECram, il faut utiliser la commande :

  $INIT/SIZE=0 device-name label

  Dans ce cas, les données présentes sur le disque DECram sont perdues

Montage d'un disque DECram :

- Il faut utiliser la commande :

  $MOUNT/NOCACHE/NOMOUNT_VERIF

- Remarque IMPORTANTE : les données sur le disque DECram sont préservées après un couple d'opération ($DISMOUNT, MOUNT)

Utilisation d'un disque DECram

- Il y a 2 types principaux d'application pouvant bénéficier de DECram :
  - — Les applications qui utilisent fréquemment des fichiers exécutables système ou des fichiers exécutables applicatifs bien définis (*read only*)
  - — Les applications modulaires qui génèrent des fichiers temporaires pouvant être recréés facilement

Fichiers exécutables :

- Il est possible d'identifier les fichiers exécutables système ou applicatifs fréquemment utilisés en utilisant des outils tels que SPM ou VPA pour identifier :
  - — Les *hot files*

— Les images fréquemment activées

— Les images générant un taux de fautes de pages élevé

- L'utilitaire VMS $INSTALL peut être utilisé pour lister les images installées fréquemment activées en utilisant la commande : $INSTALL LIST/FULL et en notant la valeur du champ "*Entry access count* " qui donne une idée du nombre de fois où ces images ont été utilisées
- Il peut être utile de placer ces fichiers exécutables ainsi que leurs librairies partageables accociées sur un disque DECram

  — Si le système est arrêté pour une raison ou une autre, il faut recopier à chaque fois ces images depuis le disque système vers le disque DECram

  — Ce type d'utilisation est très recommandé, sous réserve de disposer d'assez

de mémoire, sur une station dans un *cluster* :

> Ceci permet de faire en sorte que la pagination et donc l'activation d'image soit effectuee localement et non *re-mote*

> Ceci permet d'épargner la bande pas-sante Ethernet en conséquence

- L'installation de DECram crée une pro-cédure de commande :
  SYS$EXAMPLES:RAMDISK$STARTUP.COM
  couplée à 1 fichier de données :
  SYS$EXAMPLES:RAMDISK$NODE.DAT
  permettant de :

  — Créer le disque DECram avec une bonne taille (en fonction du fichier de données SYS$STARTUP:RAMDISK$"nom_du_noeud".DAT) si cela est possible; le disque créé par défaut est le disque MDA0:

  — Monter le disque DECram

  — Créer les *directories* nécessaires

— Ajouter le disque DECram au milieu de la *search-list* de SYS$SYSROOT

— Copier les images et librairies partageables dont la liste est donnée dans le fichier de données sur le disque DECram

- Pour utiliser cette procédure, il faut :

— Créer le fichier SYS$STARTUP:RAMDISK$"nom_du_noeud".DAT en fonction des ses besoins (analyse des résultats de VPA, SPM, $INSTALL LIST) à partir du fichier contenu dans SYS$EXAMPLES

— Modifier le fichier SYS$STARTUP:RAMDISK$"nom_du_noeud".DAT pour adapter à ses besoins les 2 paramètres (symboles) :

> extra_disk_blocks : permet de spécifier combien d'espace libre doit être laissé sur le disque DECram (après avoir créé les *directories* et copié les fichiers de la liste)

> minimum_free_memory : permet de s'assurer que le disque DECram ne sera pas créé si l'on ne dispose pas d'au moins "mimimum_free_memory" pages

— Invoquer la procédure SYS$STARTUP:RAMDISK$STARTUP.COM avec l'option "CREATE" :

@SYS$STARTUP:RAMDISK$STARTUP CREATE

Fichiers temporaires :

- Il est également possible d'améliorer les performances de son système en utilisant un disque DECram pour rediriger SYS$SCRATCH et donc rediriger la création par les images ou images partageables de VMS des fichiers temporaires vers un disque DECram

- Voici une liste partielle des images VMS créant des fichiers temporaires dans SYS$SCRATCH :

DECchart	SYS$SYSTEM:CMS.EXE
DECpresent	SYS$SYSTEM:CONVERT.EXE
SYS$LIBRARY:ADARTL.EXE	SYS$SYSTEM:EDT.EXE
SYS$LIBRARY:CMS$$MAILSHRP.EXE	SYS$SYSTEM:DTM.EXE
SYS$LIBRARY:CONVSHR.EXE	SYS$SYSTEM:MAIL.EXE
SYS$LIBRARY:EDTSHR.EXE	SYS$SYSTEM:NOTES.EXE
SYS$LIBRARY:MAILSHRP.EXE	SYS$SYSTEM:PCA.EXE
SYS$LIBRARY:NOTES$$MAILSHRP.EXE	SYS$SYSTEM:SORTMERGE.EXE
SYS$LIBRARY:PPLSHR.EXE	SYS$SYSTEM:VERIFY.EXE
SYS$LIBRARY:SMGSHR.EXE	SYS$SYSTEM:VMSTAILOR.EXE
SYS$LIBRARY:SORTSHR.EXE	

- Pour faire en sorte que les fichiers temporaires contenus dans SYS$SCRATCH soit accédés depuis le disque DECram :
  - — Créer une *directory* pour chacun des utilisateurs concernés (avec le bon UIC) : $CREATE/DIRECTORY MDA1:[user]
  - — Redéfinir SYS$SCRATCH pour pointer vers cette *directory* : $DEFINE SYS$SCRATCH MDA1:[user]
- L'installation de DECram crée une procédure de commande : SYS$EXAMPLES:SCRATCHRAM$STARTUP.COM couplée à 1 fichier de données : SYS$EXAMPLES:SCRATCHRAM$NODE.DAT permettant de :
  - — Créer le disque DECram avec une taille donnée dans le paramètre P1 (en fonction du fichier de données SYS$STARTUP:SCRATCHRAM$"nom_du_noeud".DAT) si cela est possible; le disque créé par défaut est le disque MDA1:

— Monter le disque DECram

— Créer les *directories* nécessaires pour chacun des utilisateurs listés dans le fichier :
SYS$STARTUP:SCRATCHRAM$"nom_du_noeud".DAT

- Cette procédure SYS$STARTUP:SCRATCHRAM$STARTUP admet un paramètre (P1) indiquant la taille en *blocks* du disque DECram à créer
- Chaque utilisateur doit ajouter une redéfinition de SYS$SCRATCH (table JOB) dans son fichier de "login" (peut être effectué dans le fichier de "sylogin")

Gestion système
Divers

- DECram s'installe sur toute version de VMS de V5.2 à V5.4-3
- Sur les versions de VMS de V5.2 à V5.3-2, un exécutable spécifique : SYS$SYSTEM:DECRAM$SIZE.EXE est installé permettant d'implanter la commande $INIT/SIZE=n
- Sur les versions de VMS à partir de V5.4, cette commande $INIT/SIZE=n est implantée en standard
- La commande $SHOW DEVICE/FULL ne supporte pas les disques de type DECram (DT$_RAM_DISK) et retourne un "*device type unknown*") avant VMS V5.5

- L'installation de DECram crée dans SYS$EXAMPLES 5 fichiers :
  - DECRAM$CONFIG.COM :
    - > Permet de configurer, à l'aide d'un menu, un disque DECram : création, initialisation
    - > Peu utile
  - RAMDISK$STARTUP.COM :
    - > Couplé au fichier SYS$STARTUP:RAMDISK$"nom_du_noeud".DAT
    - > Utilise le paramètre P1 pour décrire la fonction utilisée :
      - + "CREATE" :
        - * On parcourt le fichier de données pour savoir si les ressources système sont suffisantes pour créer le disque
        - * Création, initialisation et *mount* du disque MDA0:

* Création des *directories* nécessaires en fonction des informations contenues dans le fichier de données
* Modification de la *search list* de SYS$SYSROOT pour ajouter le disque DECram au milieu de la *search list*
* Copie des fichiers listés dans le fichier de données : si ces fichiers étaient précedemment installés, la procédure les réinstalle depuis le disque DECram
* Copie du fichier de données sur le disque DECram : MDA0:[000000]RAMdisk.dat
* Création d'un fichier statistique MDA0:[000000]RAMdisk$stats.dat pour enregistrer la date de création du disque DECram ainsi que le compte d'opération disque sur le

disque système et sur le disque DECram

+ "DELETE" :
  * Redéfinit la *search list* de SYS$SYSROOT pour faire pointer le disque DECram en dernier
  * Ouverture du fichier MDA0:[000000]RAMdisk.dat
  * Parcours du fichier pour supprimer et/ou désinstaller les fichiers contenus sur le disque DECram
  * Suppression complète du disque DECram de la *search list* de SYS$SYSROOT
  * *Dismount* du disque DECram et initialisation de celui-ci
+ "HITRATE" :
  * Permet de donner une idée de l'efficacité du disque DECram en comparant le nombre d'opérations

(opcnt) effectuées sur le disque système et celui effectuées sur le disque DECram

* Les comparaisons sont effectuées en utilisant le fichier de statistique créé lors de la commande "CREATE"

+ "SIZE" :

* Permet de déterminer s'il y a suffisamment de ressources système (mémoire libre, SPTEs, *global pages*, *global sections*) en fonction des informations contenues dans le fichier de données, en calculant le nombre de pages/sections globales consommées et en estimant en conséquence la taille de l'*index file* du disque DECram
* Cette opération est non destructrice et suggère comment corriger

le problème de ressources s'il y en a un

> Ce fichier doit être invoqué au moment du démarrage de la machine (base de données SYSMAN) avec le paramètre "CREATE"

— RAMDISK$NODE.DAT :

> Fichier de données d'exemple utilisé par RAMDISK$STARTUP.COM

> Utilisé pour identifier quelles *directories* et quels fichiers doivent être copiés sur le disque DECram et où ces fichiers doivent être copiés

> Exemple :

```
! Name:         RAMDISK$node.DAT
! Description:
!
! This is  the data file used by RAMDISK$STARTUP.COM during a CREATE or a
! SIZE function to build a read-only RAMdisk.
!
! The data file  must be named SYS$STARTUP:RAMDISK$node.DAT where node is
! your system's nodename.
!
! The data file is  used to identify which directories and files you wish
! to copy to the RAMdisk  and  use from that location.  Define the system
! logical  from  the  origin  of the  files  followed  by  the  filename.
! Wildcards are NOT allowed.
!
! If any of the files copied were  installed  on  the physical disk, they
! are de-installed and installed on the RAMdisk so  image activations and
! hard faults occur on the RAMdisk.
!
! CAUTION:   Images such as DCLTABLES.EXE should never be  moved  to  the
! RAMdisk  because  some  processes  (for  example,  SMISERVER) will have
! already started using the physical disk version by the time you get the
! RAMdisk loaded and running.
!
! There are also  two  parameters  examined  by the procedure.  These are
! "minimum_free_memory" and "extra_disk_blocks".  The "extra_disk_blocks"
! allows you to specify how  much  free  space  you  need on the RAMdisk.
! Extra space on the disk would  allow  you  to manually add files to the
! RAMdisk.  The "minimum_free_memory" parameter ensures the  RAMdisk will
! not be created unless it will leave at  least that amount of free pages
! in memory.  If you do not wish to  specify these parameters, simply set
! them to zero.
!

! Here are the user defined parameters:

minimum_free_memory=5000
extra_disk_blocks=0


sys$system:acmsadu.exe
sys$system:authorize.exe
sys$system:backup.exe
sys$system:cdo.exe
sys$system:cdu.exe
sys$system:decw$bookreader.exe
sys$system:decw$calc.exe
sys$system:decw$calendar.exe
sys$system:decw$cardfiler.exe
.......
```

```
!sys$library:dcltables.exe      ** do not put this file on the RAMdisk
sys$library:debugshr.exe
sys$library:decw$dwtlibshr.exe
sys$library:decw$loginout.exe
sys$library:decw$mailshr.exe
sys$library:decw$terminalshr.exe
sys$library:decw$transport_common.exe
sys$library:decw$transport_lat.exe
sys$library:decw$xlibshr.exe
sys$library:mailshr.exe
sys$library:nm_mailshr.exe
sys$library:notes$dwshare.exe
sys$library:notes$share.exe
sys$library:secureshrp.exe
sys$library:vaxcrtl.exe
sys$message:decw$dwtmsg.exe
sys$message:decw$terminalmsg.exe
sys$message:decw$transportmsg.exe
sys$message:decw$xlibmsg.exe


! Leave this  marker  here.  It is used to cleanly locate the end of this
! data file so several routines can use it.

end_of_file
```

— SCRATCHRAM$STARTUP.COM :

- > Couplé au fichier SYS$STARTUP:SCRATCHRAM$"nom_du_noeud" .DAT
- > Permet de créer un environnement de fichiers temporaires pour les utilisateurs
- > Utilise le paramètre P1 pour décrire la taille du disque DECram à créer

- > Le disque créé s'appelle MDA1:
- > Le fonctionnement de la procédure a été documenté plus haut

— SCRATCHRAM$NODE.DAT :

- > Fichier de données d'exemple utilisé par SCRATCHRAM$STARTUP.COM
- > Utilisé pour identifier quels utilisateurs doivent avoir une *directory* créée sur le disque DECram. Le nom de la *directory* est le même que celui de l'utilisateur
- > Exemple :

```
!
!
! Name:         SCRATCHRAM$node.DAT
! Description:
!
! SYS$STARTUP:SCRATCHRAM$STARTUP.COM  uses this data file  and expects it
! to be named SYS$STARTUP:SCRATCHRAM$node.DAT where node is your system's
! nodename.
!
! This is the data file listing all the users who want a directory on the
! ScratchRAM.  The most typical use  would  be  for  the  user to  assign
! SYS$SCRATCH  ScratchRAM:[user]   to  increase  temporary  file creation
! speed.
user=jones
user=haskell
! Leave this marker here.  It is used  to  cleanly locate the end of this
! data file so several routines can use it.

end_of_file
```

- Quelques idées de *tuning* liées à l'utilisation de DECram :
  - — DECram utilise la *freelist* pour y placer ses pages : ceci peut donc affecter le taux de pagination et de *swapping* car la mémoire disponible devient plus faible; il faut donc surveiller les performances de sa machine et utiliser l'*autotuning* de VMS V5.4-3
  - — Il peut donc être nécessaire de modifier les paramètres SYSGEN : FREELIM, GROWLIM, FREEGOAL
  - — Les demandes de réinstallation d'images sur le disque DECram peuvent générer une augmentation des demandes en GBLPAGES, GBLSECTIONS

- Les impacts potentiels de DECram :
  - — DECram utilise le CPU pour déplacer les données : ce CPU est cosommé dans un contexte système à niveau d'IPL élevé
  - — Une application qui était en attente d'I/O avant l'installation de DECram peut se transformer en une application très consommatrice en CPU car :
    - > DECram consomme du CPU
    - > Il n'y a PLUS d'attente d'I/O
  - — De façon générale, l'installation de DECram peut modifier de façon considérable le profil d'utilisation CPU, mémoire, I/O de son système : il faut en tenir compte

Comment ça marche ?

- DECram est implanté sous la forme d'un *driver* MDDRIVER permettant de créer un pseudo-disque en mémoire dont les pages sont allouées depuis la *freelist*

Géométrie du disque créé :

- 1 secteur par piste
- 8 pistes par cylindre
- Nombre de cylindres = taille du disque/8 pistes
- Taille maximum du disque : 256 Mbytes
  - — Cette limitation provient d'une impossibilité pour les routines LDR$ALLOC_PT et LDR$DEALLOC_PT de s'allouer plus de 524288 SPTEs
  - — Cette limitation est levée avec VMS V5.4 (patch pour V5.2 à V5.3-2 dans Télépatch)

- Taille maximum d'un transfert disque : 127 *blocks* (limitation de MOVC3)

Opérations :

- Le *driver* supporte les opérations suivantes :
  - — UNLOAD : déchargement du *driver* s'il n'y a plus de PFNs allouées au disque (permet une opération de type SYSGEN RELOAD)
  - — CONTROLLER INIT : cette routine est présente pour assurer la compatibilité avec d'autres *drivers* mais ne fait rien car on n'a pas de CSR (le champ CSR de l'IDB est remis à 0)
  - — UNIT INIT :
    - > Positionnement de l'*online bit* dans l'UCB
    - > Allocation des SPTEs permettant de *mapper* les *buffers* utilisateurs pour un transferts (128 SPTEs)

> Initialisation partielle de ces SPTEs (adresse système, PTE valide, *kernel write*)

— START I/O : comme il n'y a pas de vrai périphérique impliqué, toute l'opération d'I/O concernée est effectuée ici. Ceci implique que toute l'opération d'I/O est effectuée à *fork IPL* (ici IPL 8)

— PACKACK : positionnement du *volume valid bit* dans le *status* de l'UCB

— UNLOAD/AVAILABLE : comme on ne peut "arrêter la rotation des disques DECram", ces 2 fonctions sont équivalentes et mettent à 0, le *volume valid bit* dans le *status* de l'UCB

— FORMAT : positionne la taille du disque c'est à dire alloue ou désalloue la mémoire associée avec le disque DECram

> Calcul du nombre de cylindres, de secteurs par piste

> Appel de LDR$ALLOC_PT ou de LDR$DEALLOC_PT pour s'allouer ou désallouer les SPTs concernées

— READ : READLBLK : *read logical block*, READPBLK : *read physical block*, READ-VBLK : *read virtual block*

> Ces opérations ne sont différentes qu'en ce qui concerne le traitement des arguments d'entrée : pour une *physical I/O*, par exemple, les informations de type cylindre, piste et secteur sont converties en un *byte offset* depuis le début du *buffer* mémoire

> *Byte offset = Sector size * ((Sectors per track)*Track + Sector)*

> Ces opérations sont effectuées au moyen d'un MOVC3 dans un contexte système à IPL 8 (voir exemple plus loin)

— WRITE : WRITELBLK : *write logical block*,
WRITEPBLK : *write physical block*,
WRITEVBLK : *write virtual block*

> Ces opérations ne sont différentes qu'en ce qui concerne le traitement des arguments d'entrée : pour une *physical I/O*, par exemple, les informations de type cylindre, piste et secteur sont converties en un *byte offset* depuis le début du *buffer* mémoire

> *Byte offset = Sector size * ((Sectors per track)*Track + Sector)*

> Ces opérations sont effectuées au moyen d'un MOVC3 dans un contexte système à IPL 8

Exemple d'une lecture :

MOVC3 ByteCnt, RAMdisk_VA, Userbuffer_VA

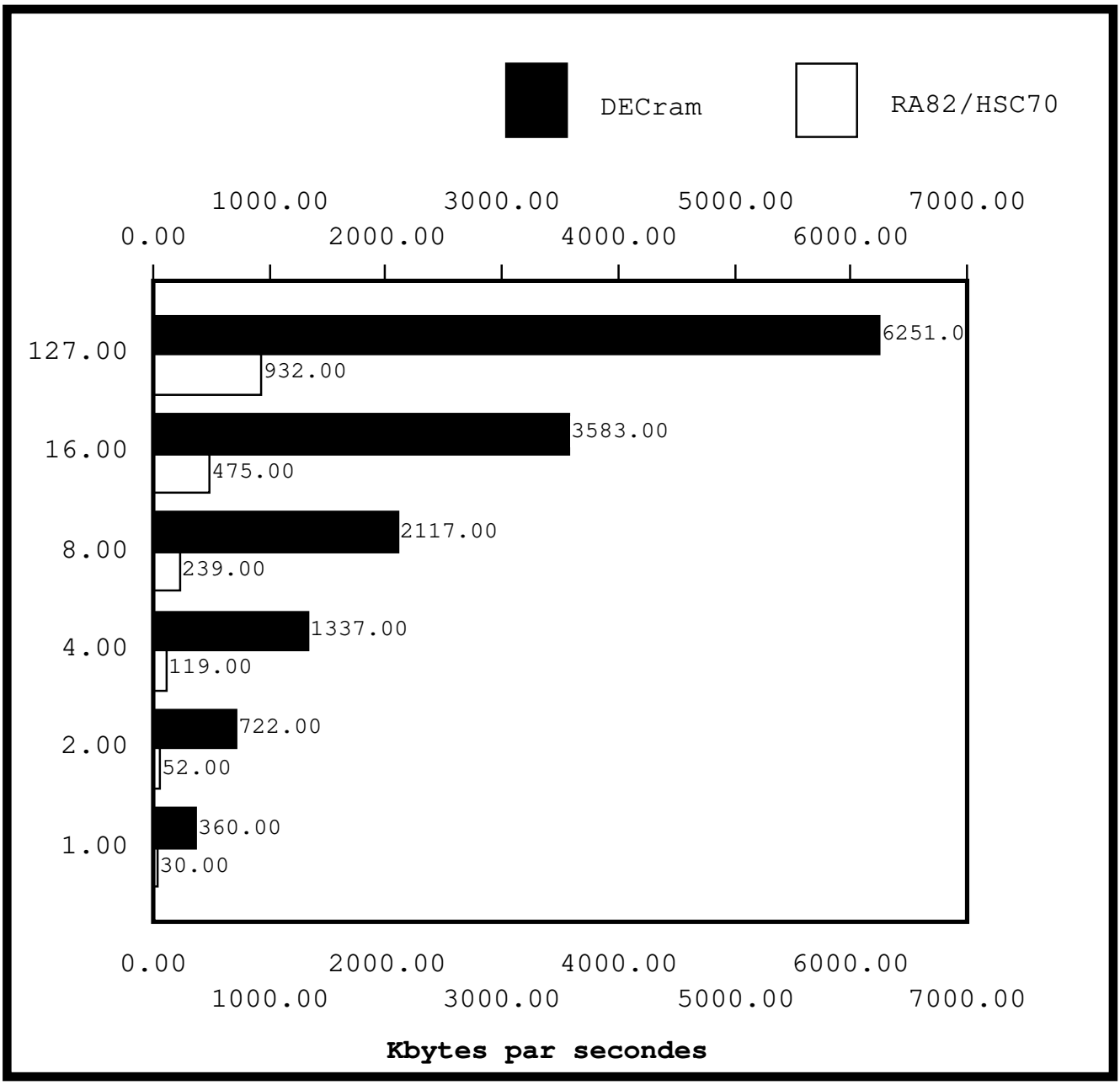
1. **RAMdisk_VA = UCB$L_MEMDISK + Byte Offset**
   a. UCB$L_MEMDISK est l'adresse virtuelle de base du disque DECram (stocké en fin d'UCB)
   b. Le *byte offset* est calculé en fonction de la formule citée ci-dessus
2. **ByteCnt = IRP$L_BCNT**
   a. Validation du fait que l'on ne cherche pas à écrire après la fin du disque
   b. Validation de la taille du transfert (<128 pages, limite de MOVC3)
3. **Map du disque DECram** :
   a. Sauvegarde de P0BR et P0LR
   b. Remplacement de P0BR et P0LR avec les valeurs de base et de longueur des PTEs privées du disque DECram

c. Invalidation du *translation Buffer* (TBIS) pour toutes les pages référencées du disque DECram

4. **Map du *buffer* utilisateur dans l'espace S0**
   a. Calcul de l'adresse virtuelle du *buffer* utilisateur
   b. Invalidation du *translation buffer* (TBIS) pour toutes les pages référencées du *buffer* utilisateur
5. **MOVC3**
6. **Restauration du contexte** :
   a. Restauration des P0BR et P0LR sauvegardées précédemment
   b. Invalidation du *translation buffer* (TBIS) pour toutes les pages référencées

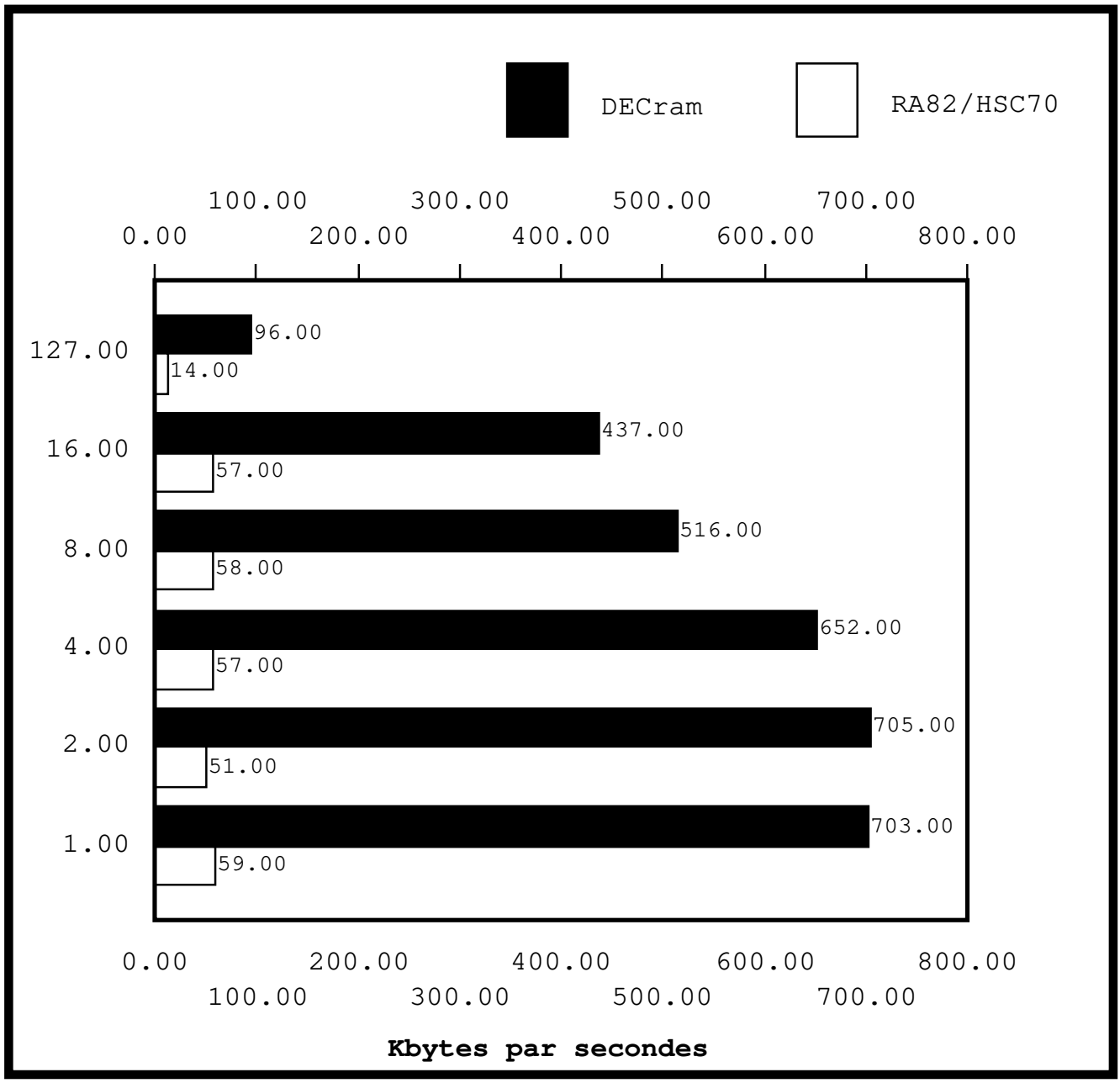
Performances

RA82/HSC70 et DECram



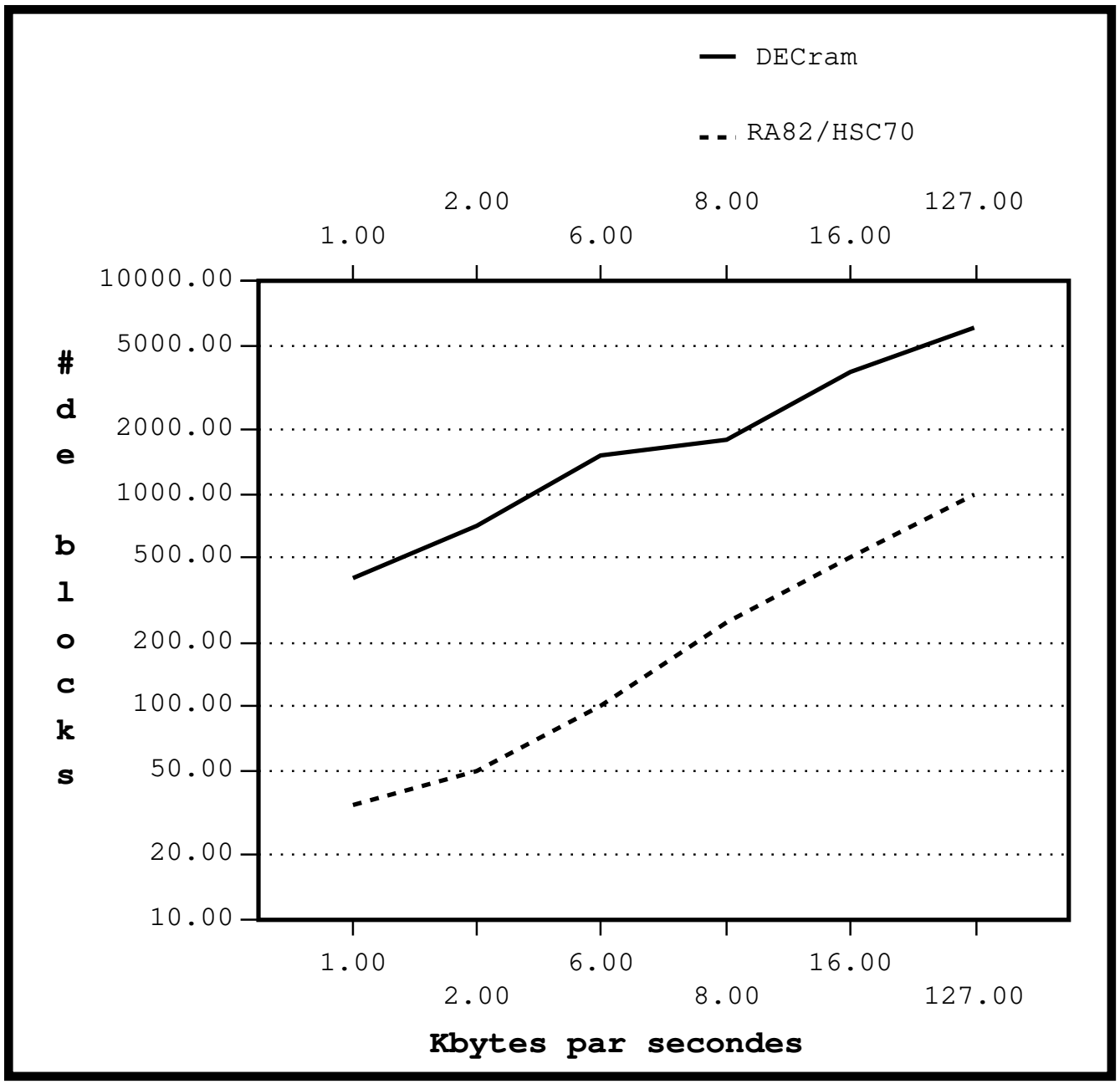
VAX 6000 modèle 6430

Performances
RA82/HSC70 et DECram



VAX 6000 modèle 6430

Performances
RA82/HSC70 et DECram



VAX 6000 modèle 6430

Performances
Applications

2 benchmarks "classiques" ont été utilisés pour mesurer les performances de DECram

- Sur les plateformes *standalone* suivantes :
  - — VAX 4000 modèle 300 (6 RF72s sur un contrôleur intégré, 128 MB, VMS V5.4-1)
  - — VAX 6000 modèle 510 (6 RF72s sur KFMSA, 384 MB, VMS V5.4-1)
  - — VAX 9000 modèle 410 (6 RA90s sur KDM70, 256 MB, VMS V5.4-2)
- Avec l'environnement logiciel suivant :
  - — VAX C V3.2
  - — VAX Cobol V4.4
  - — VAX 9000 modèle 410 tournant VMS V5.4-2

- Test 1 : CHEMPAK
  - — Logiciel de chimie moléculaire écrit en FORTRAN créant 4 fichiers *scratch* pour un total de 10000 *blocks*
  - — Application très gourmande en I/Os
  - — Résultats :
    - > VAX 9000 modèle 410 :

Configuration	Temps CPU	Temps passé	CPU en %	Amélioration
Disque magnétique	58	570	10	
DECram	59	191	31	3x

    - > VAX 6000 modèle 510 :

Configuration	Temps CPU	Temps passé	CPU en %	Amélioration
Disque magnétique	121	503	24	
DECram	123	222	55	2.3x

> VAX 4000 modèle 300 :

Configuration	Temps CPU	Temps passé	CPU en %	Amélioration
Disque magnétique	214	591	10	
DECram	218	308	31	1.9x

- Test 2 : GCC
  - — Compilateur C GNU version 1.35 distribué par la *Free Software Foundation*
  - — Mesure du temps passé pour traiter 19 fichiers sources et générer de l'assembleur SUN-3
  - — 15000 blocks alloué au disque DECram pour y placer l'exécutable et les 19 fichiers source

— Résultats :

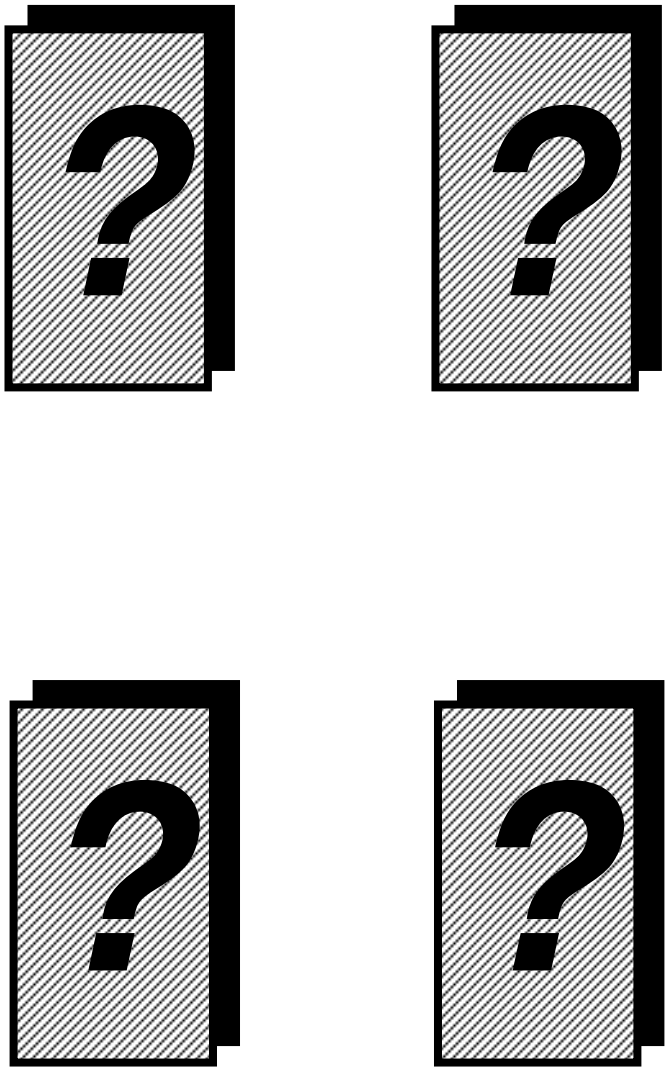
> VAX 9000 modèle 410 :

Configuration	Temps passé	Amélioration
Disque magnétique	133	
DECram	47	2.8x

> VAX 6000 modèle 510 :

Configuration	Temps passé	Amélioration
Disque magnétique	188	
DECram	99	1.9x

> VAX 4000 modèle 300 :

Configuration	Temps passé	Amélioration
Disque magnétique	243	
DECram	176	1.4x

Questions