Documentation Outil de pack IPSL

Objectif :

Transférer sur le cccstoredir et le cccworkdir les simulations stockées sur le dmnfs. Nous voulons que le format de stockage à l'arrivée soit conforme au nouveau format mis en place pour la production sur ces disques. Pour cela, l'IPSL a développé un outil de traitement des données dmnfs. Cet étape se réalise en 2 sous-étapes : la création des listes de fichiers à traiter à une certaine fréquence d'une part et le traitement de ces listes de fichiers d'autre part.

Attention, cet outil fonctionne uniquement lorsque les répertoires des données d'entrée et les répertoires de données de sortie sont situés sur le même compte utilisateur.

Rappel des différents types de fichiers créés par une simulation :

- Output au format netcdf
  - Ancien format : copiés mensuellement dans les répertoires JobName/COMP/Output/*/
  - Nouveau format : concaténés (ncrcat) par période dans les répertoires JobName/COMP/Output/*/
- Restart au format netcdf
  - Ancien format : copiés mensuellement dans les répertoires JobName/COMP/Restart/
  - Nouveau format : archivés (tar) par période dans les répertoires JobName/RESTART/ (attention on renomme les fichiers)
- Debug au format texte
  - Ancien format : copiés mensuellement dans les répertoires JobName/COMP/Debug/
  - Nouveau format : archivés (tar) par période dans les répertoires JobName/DEBUG/ (attention on renomme les fichiers)
- Post-traitements : il en existe deux types, les Analyses qui seront copiées telles quelles sur le $CCCSTOREDIR et les Monitoring et les Atlas qui seront eux copiés sur le $CCCWORKDIR

.

1ère partie : création des listes de fichiers à traiter

Variables utilisées dans cette documentation :

JobName : nom d'une simulation

DEM_utilities.sh

Ensemble de fonctions utilisées dans les scripts du Pack IPSL, permettant entre autre de gérer la création de log.

launch_ipsl_pack.sh

C'est le script principal de lancement de l'outil permettant la création des listes des fichiers à packer.

Voici la liste des scripts dans leur ordre d'appel :

- create_listing.sh
- find_directory_simul.sh
- create_config_card.sh
- calcul_size_simul.sh
- find_size_pack.sh
- write_liste_pack.sh
- archive_restart.sh
- archive_debug.sh

Ce script prend en argument d'entrée un fichier texte contenant le(s) path(s) d'un répertoire de type IGCM_OUT/ ou d'un sous-répertoire contenant une à plusieurs simulations.

Exemple de lancement :

>> ./launch_ipsl_pack.sh param_AC.txt

avec

>> vi param_AC.txt

>>> /dmnfs11/cont003/p86cozic/IGCM_OUT

Dans les scripts ce fichier (dans notre exemple param_AC.txt) est nommé ${LISTE_SIMUL}

Répertoires de travail :

L'ensemble des scripts crée un répertoire IGCM_DEM/IGCM_OUT/... dans lequel nous stockerons les listes créées pour chaque simulation étudiée. Et un répertoire IGCM_DEM/tmp/ dans lequel nous travaillons.

create_listing.sh

Ce script prend en argument d'entrée ${LISTE_SIMUL}

Pour pouvoir travailler en interrogeant un minimum l'archive nous avons décidé de créer dès le début de l'exécution un listing des fichiers existants sous le(s) path(s) pour lesquels nous voulons appliquer le Pack. Nous exploiterons constamment par la suite ce listing. Il comprend trois colonnes

- 1ère colonne : type de fichier (directory – file – link)
- 2ième colonne : taille du fichier
- 3ième colonne : path complet du fichier

Le listing créé est $SCRATCHDIR/IGCM_DEM/Listing.txt. Dans les scripts il est définit pas la variable ${LISTE_DMNFS}.

Dans notre exemple les premières lignes en sont :

d 130 /dmnfs11/cont003/p86cozic/IGCM_OUT

d 129 /dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1

d 86 /dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_06

d 45 /dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_06/ATM

d 15 /dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_06/ATM/Output

d 4096 /dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_06/ATM/Output/MO

f 19419800 /dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_06/ATM/Output/MO/ESM_06_19900101_19900130_1M_histmth.nc

f 19419800 /dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_06/ATM/Output/MO/ESM_06_19900201_19900230_1M_histmth.nc

f 19419800 /dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_06/ATM/Output/MO/ESM_06_19900301_19900330_1M_histmth.nc

f 19419800 /dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_06/ATM/Output/MO/ESM_06_19900401_19900430_1M_histmth.nc

f 19419800 /dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_06/ATM/Output/MO/ESM_06_19900501_19900530_1M_histmth.nc

f 19419800 /dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_06/ATM/Output/MO/ESM_06_19900601_19900630_1M_histmth.nc

f 19419800 /dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_06/ATM/Output/MO/ESM_06_19900701_19900730_1M_histmth.nc

f 19419800 /dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_06/ATM/Output/MO/ESM_06_19900801_19900830_1M_histmth.nc

f 19419800 /dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_06/ATM/Output/MO/ESM_06_19900901_19900930_1M_histmth.nc

Traitement des cas particuliers :

- Nous retirons directement du listing les fichiers contenus dans des répertoires SPIN/ qui seront traités ultérieurement par le CCRT.
- Nous retirons les fichiers qui sont des liens (repérés par un « l » en première colonne)
- Nous retirons les fichiers de type run.card qui seront traités ultérieurement par le CCRT.

find_directory_simul.sh

Ce script prend en argument d'entrée le fichier ${LISTE_SIMUL} et ${LISTE_DMNFS}.

Il permet pour chacun des paths inscrits dans ${LISTE_SIMUL} de lister l'ensemble des simulations situées en dessous dans l'architecture. Nous considérons que nous avons une simulation si il y a au moins un répertoire Restart/ dedans.

Cette nouvelle liste est disponible dans le fichier $SCRATCHDIR/IGCM_DEM/liste_simul_${LISTE_SIMUL_NAME}

avec

LISTE_SIMUL_NAME=$(basename ${LISTE_SIMUL})

Dans notre exemple :

$SCRATCHDIR/IGCM_DEM/liste_simul_param_AC.txt

Et les premières lignes de ce fichier sont :

/dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_06

/dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_07

/dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_08

/dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_102

/dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_Atl

/dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_200

/dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/ESM_201

/dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/CM4_200

/dmnfs11/cont003/p86cozic/IGCM_OUT/IPSL_ESM_V1/CM4_201

/dmnfs11/cont003/p86cozic/IGCM_OUT/LMDZINCA/AER/AER_REF

/dmnfs11/cont003/p86cozic/IGCM_OUT/LMDZINCA/AER/AER_REF2

/dmnfs11/cont003/p86cozic/IGCM_OUT/LMDZINCA/AER/AER_MERGE

/dmnfs11/cont003/p86cozic/IGCM_OUT/LMDZINCA/AER/AER_v2

create_config_card.sh

Ce script prend en arguments d'entrée

$SCRATCHDIR/IGCM_DEM/liste_simul_${LISTE_SIMUL_NAME} et ${LISTE_DMNFS}

Ce script permet pour chacune des simulations listées de recréer sa carte d'identité « config.card » (${IGCM_DEM_SIMU}/config_card_${JobName}). Cette carte contiendra les informations suivantes :

- JobName : nom de la simulation
- DateBegin : date de début de la simulation
- DateEnd : date de fin de la simulation
- PATH_SIMU_FULL : path de la simulation sur le dmnfs
- IGCM_DEM_SIMU : path sur le scratchdir dans lequel nous stockerons les listes.

Pour trouver DateBegin et DateEnd nous utilisons les noms des fichiers Restart et Output créés par une simulation. Chacun de ces fichiers contient dans son nom la date du mois de simulation auquel il correspond. On conserve en DateBegin la plus petite date trouvée et en DateEnd la plus grande date trouvée lorsque l'on passe en revue tous ces fichiers.

Si nous considérons la simulation

/dmnfs11/cont003/p86cozic/IGCM_OUT/LMDZORINCA/AER/AER_AR5WERF_VALID

Sa carte sera :

$SCRATCHDIR/IGCM_DEM/IGCM_OUT/LMDZORINCA/AER/AER_AR5WERF_VALID/config_card_AER_AR5WERF_VALID

et elle contiendra les informations suivantes :

JobName=AER_AR5WERF_VALID

DateBegin=20000101

DateEnd=20001231

PATH_SIMUL_FULL=/dmnfs11/cont003/p86cozic/IGCM_OUT/LMDZORINCA/AER/AER_AR5WERF_VALID

IGCM_DEM_SIMU=/scratch/cont003/p86cozic/IGCM_DEM/IGCM_OUT/LMDZORINCA/AER/AER_AR5WERF_VALID

L'ensemble des cartes créées sont listées dans le fichier $SCRATCHDIR/IGCM_DEM/config_card.liste. Il contient deux colonnes :

- 1ère colonne : pour chaque simulation contenue dans

  liste_simul_${LISTE_SIMUL_NAME} le path de sa carte config.card

- 2ième colonne : un log permettant le management des scripts. A ce stade «*ListToBeDone* »

Dans notre exemple les premières lignes de ce fichier sont

/scratch/cont003/p86cozic/IGCM_DEM/IGCM_OUT/IPSL_ESM_V1/ESM_06/config_card_ESM_06 ListToBeDone

/scratch/cont003/p86cozic/IGCM_DEM/IGCM_OUT/IPSL_ESM_V1/ESM_07/config_card_ESM_07 ListToBeDone

/scratch/cont003/p86cozic/IGCM_DEM/IGCM_OUT/IPSL_ESM_V1/ESM_08/config_card_ESM_08 ListToBeDone

/scratch/cont003/p86cozic/IGCM_DEM/IGCM_OUT/IPSL_ESM_V1/ESM_102/config_card_ESM_102 ListToBeDone

/scratch/cont003/p86cozic/IGCM_DEM/IGCM_OUT/IPSL_ESM_V1/ESM_Atl/config_card_ESM_Atl
ListToBeDone

/

Le script create_config_card permet également de créer, pour chaque simulation, les listings par type de fichiers et commence à traiter certains cas particuliers.

Les listing créés et qui seront utilisés par les scripts suivants sont :

- La liste des fichiers de restart (on conserve uniquement les colonnes 2 et 3 du Listing)

  ${IGCM_DEM_SIMU}/liste_restart_files_config.txt

Si l'on reprend notre exemple les premières lignes de cette liste sont du type :

49905924
/dmnfs11/cont003/p86cozic/IGCM_OUT/LMDZORINCA/AER/AER_AR5WERF_VALID/ATM/Restart/AER_AR5WERF_VALID_20000131_restart.nc

16040520
/dmnfs11/cont003/p86cozic/IGCM_OUT/LMDZORINCA/AER/AER_AR5WERF_VALID/ATM/Restart/AER_AR5WERF_VALID_20000131_restartphy.nc

49905924
/dmnfs11/cont003/p86cozic/IGCM_OUT/LMDZORINCA/AER/AER_AR5WERF_VALID/ATM/Restart/AER_AR5WERF_VALID_20000229_restart.nc

16040520
/dmnfs11/cont003/p86cozic/IGCM_OUT/LMDZORINCA/AER/AER_AR5WERF_VALID/ATM/Restart/AER_AR5WERF_VALID_20000229_restartphy.nc

- La liste des outputs (on conserve uniquement les colonnes 2 et 3 du Listing) :

  ${IGCM_DEM_SIMU}/liste_output_files_config.txt

Si l'on reprend notre exemple les premières lignes de cette liste sont du type

1538315256
/dmnfs11/cont003/p86cozic/IGCM_OUT/LMDZORINCA/AER/AER_AR5WERF_VALID/ATM/Output/MO/AER_AR5WERF_VALID_20000101_20000131_1M_histmth.nc

1439077304
/dmnfs11/cont003/p86cozic/IGCM_OUT/LMDZORINCA/AER/AER_AR5WERF_VALID/ATM/Output/MO/AER_AR5WERF_VALID_20000201_20000229_1M_histmth.nc

1538315256
/dmnfs11/cont003/p86cozic/IGCM_OUT/LMDZORINCA/AER/AER_AR5WERF_VALID/ATM/Output/MO/A

ER_AR5WERF_VALID_20000301_20000331_1M_histmth.nc

1488696280
/dmnfs11/cont003/p86cozic/IGCM_OUT/LMDZORINCA/AER/AER_AR5WERF_VALID/ATM/Output/MO/AER_AR5WERF_VALID_20000401_20000430_1M_histmth.nc

1538315256
/dmnfs11/cont003/p86cozic/IGCM_OUT/LMDZORINCA/AER/AER_AR5WERF_VALID/ATM/Output/MO/AER_AR5WERF_VALID_20000501_20000531_1M_histmth.nc

1488696280
/dmnfs11/cont003/p86cozic/IGCM_OUT/LMDZORINCA/AER/AER_AR5WERF_VALID/ATM/Output/MO/AER_AR5WERF_VALID_20000601_20000630_1M_histmth.nc

De ces deux listes on retire tous les fichiers qui ne sont pas du type $JobName_$date ou qui sont dans des sous-répertoires de Restart/ ou de Output/*/. Ces éventuels fichiers ne sont pas créés par libIGCM et nous ne savons pas comment les traiter. Leurs listes sont actuellement stockées dans les fichiers ${IGCM_DEM_SIMU}/other_tar/tar_no_output_files.list et

${IGCM_DEM_SIMU}/other_tar/tar_no_restart_files.list . Nous ne savons pas encore ce que nous ferons de ces listes.

Nous créons également deux répertoires ${IGCM_DEM_SIMU}/work_cp/ et

${IGCM_DEM_SIMU}/store_cp/ dans lesquels nous stockerons les listes de fichiers à copier sans rien modifier entre le dmnfs et le nouveau système de stockage. Ces listes contiennent les Monitoring, les Atlas, les fichiers d'Analyse et le répertoire Exe/. Ainsi que pour certaines simulations un fichier nommé mesh_mask.nc qui est unique et ne peut donc être packé avec rien d'autre.

Si la simulation ne contient pas de fichiers de type Output nous considérons qu'elle n'est pas traitable et nous l'archivons entièrement via tar (appelle à write_liste_tar.sh, voir documentation plus bas).

Désormais nous allons travailler sur chacune des cartes de config créées (${CONFIG})

calcul_size_simul.sh

Ce script prend en argument une carte de config ${CONFIG} et le listing ${LISTE_DMNFS}.

Si une simulation a une taille de stockage inférieure à 1Go nous décidons de l'archiver via tar sans utiliser la concaténation des sorties.

Pour cela nous appelons le script write_liste_tar.sh

write_liste_tar.sh

Ce script prend en argument d'entrée une carte de type config.card. Il crée un fichier

${IGCM_DEM_SIMU}/tar_full_simul.txt qui contient le path de la simulation à tarer. Il indique l'état *WriteListTarDone* dans le fichier $SCRATCHDIR/IGCM_DEM/config_card.liste.

find_size_pack.sh

Ce script prend en argument d'entrée une carte de config ${CONFIG} et le listing

${LISTE_DMNFS}.

Dans ce script nous travaillons avec la liste de fichiers d'Output créé précédemment :

${IGCM_DEM_SIMU}/liste_output_files_config.txt

Il permet de déterminer pour une simulation donnée la fréquence de pack optimal. Pour trouver cette fréquence nous donnons une contrainte : qu'elle soit la même pour tous les types de fichiers.

Nous commençons par définir une taille de pack idéale : entre 20 et 70 Go, et une fréquence idéale : 20 ans. Ensuite pour chaque type de fichier d'output nous vérifions si pour la fréquence idéale nous respectons la taille idéale.

Si pour la fréquence idéale la taille obtenue pour un pack est trop grande ou trop petite nous appliquons une règle de trois pour obtenir la nouvelle fréquence répondant au critère de taille idéale. Au final nous retenons la plus petite fréquence calculée quelque soit le type de fichier d'Output.

Quelques cas particuliers sont traités :

- nous n'utilisons pas la taille calculée pour la dernière période (car si la simulation a un nombre d'années qui n'est pas un multiple de 20 alors la taille de la dernière période est obligatoirement plus petite que les précédentes)
- si nous n'avons qu'une seule période (donc pour les simulations dont le nombre d'années est inférieur à 20) nous recalculons la taille de la période avant de recalculer la fréquence idéale
- Si la fréquence idéale est inférieure à 1 an nous arrondissons à 1.
- Nous ramenons quoiqu'il arrive la fréquence idéale à l'une des suivantes : 1 – 5 – 10 – 20 – 50 – 100

Au final la fréquence qui vient d'être calculée est stockée dans le fichier

${IGCM_DEM_SIMU}/period_pack.txt.

write_list_pack.sh

Comme les scripts précédents celui ci prend en arguments d'entrée un fichier de config ${CONFIG} et le listing ${LISTE_DMNFS}.

Ce script écrit les listes de fichiers d'output à concaténer, en respectant la fréquence calculée par find_size_pack.sh. Si dans une période il manque un fichier nous archivons cette période pour ce type de fichier au lieu de la concaténer. Nous créons deux répertoires

${IGCM_DEM_SIMU}/output_ncrcat/ qui contiendra les listes à concaténer avec ncrcat, et

${IGCM_DEM_SIMU}/output_tar/ qui contiendra les listes à archiver avec tar.

Les listes créées sont nommées :

${JobName}_${datebeginperiod}_${dateendperiod}_${typefichier}.list avec

- JobName : le nom de la simulation
- datebeginperiod : la date de début de la période packée
- dateendperiod : la date de fin de la période packée
- typefichier : le type de fichier d'ouput traité dans cette liste (le nom contient l'extension .nc)

archive_restart.sh

Ce script prend en arguments d'entrée une carte de config ${CONFIG} et le listing

${LISTE_DMNFS}.

Ce script va permettre de créer la liste des restarts à archiver. Il utilise la liste

${IGCM_DEM_SIMU}/liste_restart_files_config.txt créée par create_config_card.sh et la fréquence de pack calculée par find_size_pack.sh.

En plus d'être regroupé par période les fichiers de restart sont renommés et stockés dans un nouveau répertoire JobName/RESTART/.

Exemple pour les fichiers renommés Ancien stockage : JobName/ATM/Restart/JobName_ouput1.nc JobName/SRF/Restart/JobName_output2.nc Nouveau stockage :

JobName/RESTART/JobName_restart_debutperiode_finperiode.tar

avec

>> tar tf JobName_restart_debutperiode_finperiode.tar

>>ATM_JobName_output1.nc

>>SRF_JobName_output2.nc

Au final les listes de restart à archiver sont dans le répertoire ${IGCM_DEM_SIMU}/restart_tar/.

archive_debug.sh

Ce script est calqué sur le précédent mais il traite les fichiers stockés anciennement dans les répertoires Debug/ et Out/. Au final les listes créées sont stockées dans le répertoire

${IGCM_DEM_SIMU}/debug_tar/.

Résumé de l'état final après passage de l'outil

Dans le répertoire ${IGCM_DEM_SIMU} nous avons créé les directories suivant :

work_cp

>> ls work_cp

>> cp_files.list

>> vi cp_files.list

>> liste des répertoires et des fichiers à copier directement sur le cccworkdir

store_cp

>> ls store_cp

>> cp_files.list

>> vi cp_files.list

>> liste des répertoires et des fichiers à copier directement sur le cccstoredir.

restart_tar

>> ls restart_tar

>> listes, à une fréquence donnée, des fichiers de restart à archiver

debug_tar

>> ls debug_tar

>> listes, à une fréquence donnée, des fichiers de debug à archiver

output_ncrcat

>> ls output_ncrcat

>> listes, à une fréquence donnée, des fichiers d'output à concaténer

output_tar

>> ls output_tar

>> listes, à une fréquence donnée, des fichiers d'ouputs à archiver

1- parce qu'il manque un fichier dans la période.

2- parce qu'il y a eu une erreur lors de la concaténation

RESTART/

>> ls RESTART

>> liens vers les fichiers Restart de la simulation en les renommant

DEBUG/

>> ls DEBUG/

>> liens vers les fichiers Debug de la simulation en les renommant

other_tar/

>> ls other_tar

>> contient des listes de fichiers qui ne sont pas produits par libIGCM et qui seront laissés dans l'espace de démigration avant d'être traités par le CCRT.

tar_full_simul.txt

Ce fichier n'existe que si la simulation est inférieure à 1Go ou si elle ne contient pas de fichiers d'Output (simulation plantée).

2ème partie : outil de traitement des données

Objectif

A partir des listes de fichiers créées, on veut réaliser les étapes correspondantes :

- concaténation à base de la commande « nco » : ncrcat
- archivage avec la commande tar
- copie de fichiers directement vers le CCCSTOREDIR et CCCWORKDIR.

Fonctionnement

L'outil se compose d'un script principal « launch_ipsl_enlarge.sh » qui lance un script secondaire « enlarge_my_files ».

- launch_ipsl_enlarge.sh

C'est le script shell principal qui fait le lien entre les listes de fichiers à traiter et le script qui réalise les commandes.

Il se lance de la façon suivante :

```
>> ./ launch_ipsl_enlarge.sh
```

On boucle sur les simulations recensées durant l'étape de création de listes dans le fichier et répertoriées dans le fichier $SCRATCHDIR/IGCM_DEM/config.card.liste.

Pour chacune d'entre elles, le script « enlarge_my_files.sh » est lancé.

```
>> ./enlarge_my_files  ${INPUT_DMF_DATA} ${OUTPUT_STORE} ${OUTPUT_WORK}
```

avec

INPUT_DMF_DATA = path de la copie du dmnfs sur l'espace tampon

OUTPUT_STORE=path sur l'espace tampon ou seront stockés les fichiers traités et avant envoi sur CCCSTORE

OUTPUT_WORK=path sur l'espace tampon ou seront stockés les fichiers traités et avant envoi sur CCCWORK

- enlarge_my_files.sh

C'est le script shell qui va concrètement réaliser les différentes commandes de traitement sur les fichiers. On boucle sur les directories obtenus en fin d'étape de création de listes et pour chacun d'entre eux on traite les différentes listes de fichiers.

Un fichier de log et un fichier status sont crées pour chacune des listes à traiter dans le répertoire dans lequel se trouvent les listes. Les états possibles sont : COMPLETED, FAILED ou DELEGATE (dans le cas ncrcat seulement).

output_ncrcat : listes, à une fréquence donnée, des fichiers d'output à concaténer. « ncrcat » est la commande qui permet de concaténer une série de fichiers. Plus d'infos : http://nco.sourceforge.net/nco.html#ncrcat-netCDF-Record-Concatenator

On utilisera en particulier les options suivantes:

- --md5_digest , avec la version 4.10, ncrcat effectue une vérification sur la signature des variables pour chaque record, en entrée et en sortie. La commande lève une erreur s'il y a une différence.

- -x -v aaaa,bbbb,cccc Certaines variables ne se retrouvent pas dans tous les fichiers à concaténer. Il faut donc les exclure à travers une liste à construire dynamiquement via le script bash suivant

```
nbfile=0
for file in `cat $1` ; do
  ncdump -h ${file} | gawk '{if (match($0, /(byte|char|short|int|float|double) (.*)\(/, arr)) print arr[2] }' >> tmp_$$.txt
  let nbfile=nbfile+1
done
varstoexclude=`cat tmp_$$.txt | sort | uniq -c | awk -v nbfile=$nbfile '{if ($1 != nbfile) {print $2}}' | paste -s -d','`
rm -f tmp_$$.txt
```

La commande finale lancée est :

```
ncrcat --md5_digest -x -v aaaa,bbbb,cccc
```

Si la commande renvoie un code d'erreur, le statut retourné dans le fichier status est FAILED. Par défaut, 3 tentatives sont nécessaires avant de passer à l'état DELEGATE : l'archivage se fera alors via la commande tar plutôt que ncrcat et la liste de fichiers à traiter est déplacée dans le répertoire output_tar.

Si la commande ne renvoie pas de code d'erreur, le statut retourné dans le fichier status sera COMPLETED.

output_tar : listes, à une fréquence donnée, des fichiers d'output à archiver avec la commande tar en raison d'une erreur à la concaténation (via ncrcat).

On utilisera en particulier les options suivantes de la commande tar :

- -W (verify) : lorsque cette option est spécifiée, l'intégralité du tar est vérifiée et comparée au contenu des fichiers sources (octet par octet). Cette option est assez coûteuse puis qu'elle parcourt et lit deux fois les fichiers sources (une fois pour faire le tar, une autre fois pour comparer le contenu).

- --format=posix : qui génnère les tar au format POSIX.1-2001
  Ce format supprime les limitations des formats précédents, notamment : longueur des noms de fichiers, taille des fichiers, nombre de fichiers, ...

La commande finale lancée est :

```
tar --format=posix -W -cf output.tar input_dir
```

Si la commande renvoie un code d'erreur, le statut retourné dans le fichier status est FAILED. Une seule tentative est faite.

Si la commande ne renvoie pas de code d'erreur, le statut retourné dans le fichier status sera COMPLETED.

restart_tar : listes, à une fréquence donnée, des fichiers restarts à archiver.

On utilisera en particulier les options suivantes de la commande tar :

- -W (verify) : lorsque cette option est spécifiée, l'intégralité du tar est vérifiée et comparée au contenu des fichiers sources (octet par octet). Cette option est assez coûteuse puis qu'elle parcourt et lit deux fois les fichiers sources (une fois pour faire le tar, une autre fois pour comparer le contenu).
- --format=posix : qui génnère les tar au format POSIX.1-2001
  Ce format supprime les limitations des formats précédents, notamment : longueur des noms de fichiers, taille des fichiers, nombre de fichiers, ...

La commande finale lancée est :

```
tar --format=posix -W -cf output.tar input_dir
```

Si la commande renvoie un code d'erreur, le statut retourné dans le fichier status est FAILED. Une seule tentative est faite.

Si la commande ne renvoie pas de code d'erreur, le statut retourné dans le fichier status sera COMPLETED.

debug_tar : listes, à une fréquence donnée, des fichiers debug à archiver.

On utilisera en particulier les options suivantes de la commande tar :

- -W (verify) : lorsque cette option est spécifiée, l'intégralité du tar est vérifiée et comparée au contenu des fichiers sources (octet par octet). Cette option est assez coûteuse puis qu'elle parcourt et lit deux fois les fichiers sources (une fois pour faire le tar, une autre fois pour comparer le contenu).
- --format=posix : qui génère les tar au format POSIX.1-2001
  Ce format supprime les limitations des formats précédents, notamment : longueur des noms de fichiers, taille des fichiers, nombre de fichiers, ...

La commande finale lancée est :

```
tar --format=posix -W -cf output.tar input_dir
```

Si la commande renvoie un code d'erreur, le statut retourné dans le fichier status est FAILED. Une seule tentative est faite.

Si la commande ne renvoie pas de code d'erreur, le statut retourné dans le fichier status sera COMPLETED.

store_cp : liste des répertoires et fichiers à copier directement sur le CCCSTOREDIR

Une copie sera faite de $INPUT_DMF_DATA vers $OUTPUT_STORE.

La commande finale lancée est :

```
cp –rf $INPUT_DMF_DATA /input_dir $OUTPUT_STORE
```

Si la commande renvoie un code d'erreur, le statut retourné dans le fichier status est FAILED. Une seule tentative est faite.

Si la commande ne renvoie pas de code d'erreur, le statut retourné dans le fichier status sera COMPLETED.

work_cp : liste des répertoires et fichiers à copier directement sur le CCCWORKDIR

Une copie sera faite de $INPUT_DMF_DATA vers $OUTPUT_WORK.

Si la commande renvoie un code d'erreur, le statut retourné dans le fichier status est FAILED. Une seule tentative est faite.

Si la commande ne renvoie pas de code d'erreur, le statut retourné dans le fichier status sera COMPLETED.

La commande finale lancée est :

```
cp –rf $INPUT_DMF_DATA /input_dir $OUTPUT_WORK
```