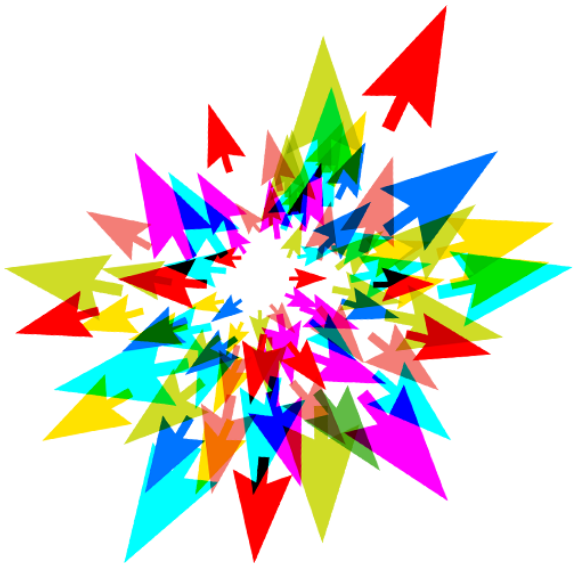
Calcul Québec

Outils de paramétrisation/lancement de tâches multiples

compute canada | calcul canada



lots de tâches (job arrays),

GNU Parallel, GLOST,

BqTools

présentation avec example

Calcul Québec

Cas classiques avec lancement des tâches multiples

traitement d’un ensemble des fichiers

variation des paramètres des fichiers d’entré

1) code (script) model / template
2) fichier d’entré model / template
3) code (application) acceptant des arguments / paramètres

Description d’un cas imaginaire

Fonction : pour un fichier “a.txt” donné, on extrait les lignes contenant une “chaine”, puis ajouter “+entier” à la fin des liges extraites ⇒ les sauvegarder dans un fichier resultat.

```
[luhuizho@ip15-mp2 EXAMPLE]$ cat travail_un_cas.sh
f=$1
search_str=$2
ajout_val=$3
case_dir=${f}___${search_str}___${ajout_val}___case
mkdir ${case_dir}
cd ${case_dir}
ln -s ../$f .
grep "${search_str}=" $f | sed -e "s/\$/+${ajout_val}/g" \
     > ${f}___${search_str}___${ajout_val}
```

Calcul Québec

Description d'un cas imaginaire – continue

Variation des 3 arguments de la fonction :

1) "a.txt" est un des fichiers du présent repertoire :

a1.txt, a2.txt …

2) "chaine" venu de chaque ligne du fichier search_str.txt

3) "entier" est soit 10, soit 20

```
[luhuizho@ip20-mp2 EXAMPLE]$ cat search_str.txt
ac
bb
[luhuizho@ip20-mp2 EXAMPLE]$ ls a*txt
a1.txt  a2.txt  a3.txt  a4.txt
[luhuizho@ip20-mp2 EXAMPLE]$ tail -2 a1.txt
ec=1
ed=2
```

Calcul Québec

Description d’un cas imaginaire – continue

à la fin, on obtient des nouveaux fichiers comme :

a1.txt___ac___10___case

⇒ chaine cherchée : “ac=” ; ajouté fin de ligne : “+10”

a1.txt___bb___10___case

a1.txt___ac___20___case

…..

Ensemble, il y a nbFichier X nbChaines X nbNumbresAjoutés cas

(produit cartésien ici … 4*2*2=16)

[luhuizho@ip15-mp2 EXAMPLE]$ cat a1.txt___ac___10

ac=1+10

Script de soumission

```
#!/bin/bash
#SBATCH --account=def-luhuizho
#SBATCH --ntasks=1        # nb de tâches
#SBATCH --mem-per-cpu=1024M     # mémoire par cpu
#SBATCH --time=0-00:10          # temps (DD-HH:MM)
….lancement d’application….
```

Calcul Québec

algorithme séquentiel (traitement simple)

```
for f in `ls a*txt` ; do
        cat search_str.txt | while read search_str ; do
                for ajout_val in 10 20 ;  do
                        travail_un_cas.sh ${f} ${search_str} ${ajout_val}
                done
        done
done
```

Calcul Québec

algorithme séquentiel – classique

(utiliser fichier d'entré model/template)

```
for f in `ls a*txt` ; do
    cat search_str.txt | while read search_str ; do
        for ajout_val in 10 20 ;  do
            cat template.txt | sed -i  \
            -e "s/~~f~~/$f/g" \
            -e "s/~~search_str~~/$search_str/g" \
            -e "s/~~ajout_val~~/$ajout_val/g" > unCas.txt
             MonApplication < unCas.txt
 Done; done ; done
```

Calcul Québec

Slurm Job Array (lots de tâches)

https://slurm.schedmd.com/job_array.html

https://docs.computecanada.ca/wiki/Job_arrays

Lancer un ensemble (vecteur) de tâches (séquentielles ou parallèles)

Chaque tâche est différentiée/identifiée par l'élément correspondant de cet ensemble/vecteur : $SLURM_ARRAY_TASK_ID

Slurm Job Array : continue-1

Calcul Québec

	$SLURM_ARRAY_TASK_ID
sbatch --array=0-7	Un des entier entre 0 et 7 (inclusive)
sbatch --array=1,3,6	Un des 3 entiers listés (1,3,6)

*** cas simple : on utilise cet identificateur comme paramètre directement (pas de nombre flottant)**

*** cas compliqué : établit une correspondance entre cet identificateur et le(s) cas à traiter par cette tâche — par example SLURM_ARRAY_TASK_ID est indice de la liste ordonnée des cas à traiter**

Slurm Job Array : continue-2

```
#SBATCH –ntasks=1
compte=0
for f in `ls a*txt` ;  do
      for  search_str in `cat search_str.txt` ;  do
            for ajout_val in 10 20 ;  do
                  compte=$((compte+1))
                  compte_modulo=$(($compte % $SLURM_ARRAY_TASK_COUNT + 1))
                  if [ "$SLURM_ARRAY_TASK_ID" -eq "$compte_modulo" ];  then
                        travail_un_cas.sh  ${f} ${search_str} ${ajout_val}
                  fi
 Done ; done ;  done
```

sbatch –array=1-8 jobArray.slurm # ⇒ job (148275)

(chaque tâche traite 2 combinaisons)

sbatch --depend=afterok:148275 jobArray_post.slurm

Calcul Québec

Gnu Parallel

https://www.gnu.org/software/parallel/man.html

https://docs.computecanada.ca/wiki/GNU_Parallel

Un outil permettant d'exécuter plusieurs tâches séquentielles en parallèle (éventuellement sur différents machines/cœurs)

Gnu Parallel : continue - 1

Calcul Québec

```
parallel echo ::: A B C
parallel echo ::: A B C ::: `seq 0.01 0.02 0.1` # A 0.01, ...
parallel echo ::: awk -F, '{print $2}' fichier.csv # 2eme colonne
parallel echo ::: A B C ::: {1..3} # A 1 ; A 2 ; A 3 ; B 1 ; ...
parallel echo ::: A B C :::+ {1..3} # A 1 ; B 2 ; C 3 ;
parallel echo :::: a.txt ::: {1..3} # ligne-1 1 ; ligne-1 2 ; …
parallel :::: cmd_file # exécuter en parallèle chaque ligne
parallel –jobs 3 --sshloginfile listDesNoeuds --workdir $PWD ….
# exécuter 3 tâches simultanément sur chaque machine

find . -name ‘*txt’ | parallel gzip
```

Écrit en perl : beaucoup de fonctionnalité de traiter les text

Gnu Parallel : continue - 2

Calcul Québec

```
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=4

scontrol show hostname $SLURM_JOB_NODELIST  > node_list_file
parallel --joblog example.log \
         --jobs $SLURM_NTASKS_PER_NODE \
        --sshloginfile node_list_file \
         --workdir $PWD \
          ./travail_un_cas.sh ::: a*txt :::: search_str.txt ::: 10 20
# reprise des cas échoués, option: --resume-failed --joblog example.log
# --env MyEnv , permet d'exporter la variable d'environnement
```

Gnu Parallel : continue – 4

(utilisé dans job array)

```
#SBATCH –ntasks=1

parallel -k echo './travail_un_cas.sh' \
 {= "if(seq()%${SLURM_ARRAY_TASK_COUNT}!=${SLURM_ARRAY_TASK_ID}-1) { skip() }" =}  \
::: a*txt :::: search_str.txt ::: 10 20 \
2>/dev/null > .tmp_$SLURM_ARRAY_TASK_ID

bash .tmp_$SLURM_ARRAY_TASK_ID
#on remplace la boucle du script original de Job-Array par gnu-parallel !
#les commandes/applications peuvent être différentes
```

sbatch –array=1-8 simp_jobArrayAvecGnuPar.slurm

Calcul Québec

GLOST

(Greedy Launcher Of Small Tasks)

https://github.com/cea-hpc/glost

https://docs.computecanada.ca/wiki/GLOST

un outil pour exécuter un grand nombre de tâches séquentielles de courte durée ou de durée variable

(utiliserser mpiexec/srun de façon maître-esclave)

(facile de lancer des applications différentes)

GLOST : continue

Calcul Québec

```
#SBATCH --job-name=HLU_TEST
#SBATCH --nodes=2
#SBATCH --ntasks-per-node=4
#SBATCH --mem-per-cpu=1000M
#SBATCH --time=5-00:00
#SBATCH --account=def-luhuizho
module load  glost/0.3.1
parallel echo ./travail_un_cas.sh  ::: a*txt :::: search_str.txt ::: 10 20 > exe_tous
srun glost_launch exe_tous
```

```
[luhuizho@ip15-mp2 EXAMPLE]$ cat exe_tous
./travail_un_cas.sh a1.txt ac 10
./travail_un_cas.sh a1.txt ac 20
./travail_un_cas.sh a1.txt bb 10
....
```

Calcul Québec

BqTools
(Batch Queueing Tools)

Outils pour simplifier la soumission d'un grand nombre de tâches sur cluster : même commande/application, différents ensembles de données.

Mise à jours sur MP2b (v5.0-RC1) : implémenté sur Slurm job-array et GNU-Parallel. Chaque tâche s'exécute dans son propre répertoire avec instance de fichier d'entré construit en fonction des valeurs définies dans le fichier d'entré global de BqTools.

Tâches incomplètes/annulées peuvent être redémarrées avec bqsubmit

Calcul Québec

BqTools : continue – 1

(script de soumission, fichier template)

```
[luhuizho@ip15-mp2 EXAMPLE]$ cat bqsubmit.dat
concurrentJobs=8
command=bash travail_un_cas.sh
templateFiles=un_cas.sh
param1=f=$(ls a*txt)
param2=search_str=$(cat search_str.txt)
param3=ajout_val=$(seq 10 10 20)
submitOptions="--account=def-luhuizho "
[luhuizho@ip15-mp2 EXAMPLE]$ cat un_cas.sh
dir=/home/luhuizho/FORMATION/Parametrisation/BqTools/EXAMPLE
egrep "~~search_str~~="$dir/~~f~~ \
 | sed -e "s/\$/+~~ajout_val~~/g" \
> ~~f~~___~~search_str~~___~~ajout_val~~
[luhuizho@ip15-mp2 EXAMPLE]$ bqsubmit
```

Calcul Québec

BqTools : continue – 2
(structure des répertoires resultats)

[luhuizho@ip15-mp2 EXAMPLE]$ ls | egrep BQ

BATCH_00001.BQ, BATCH_00002.BQ, ….. # sous-repertoire crée par bqsubmit

BATCH-389948_1.out, BATCH-389948_2.out, … # output de chaque tâche

BATCH_fa1.txt_search_strac_val_to_add10.BQ, … # lien symbolique du sous-sous-repertoire de résultat

[luhuizho@ip16-mp2 EXAMPLE]$ ls BATCH_00001.BQ

00001.BQ logfile

[luhuizho@ip16-mp2 EXAMPLE]$ ls BATCH_00001.BQ/00001.BQ/

a1.txt___ac___10 travail_un_cas.sh

[luhuizho@ip16-mp2 EXAMPLE]$ cat BATCH_00001.BQ/00001.BQ/travail_un_cas.sh

egrep "ac=" /home/luhuizho/FORMATION/Parametrisation/BqTools/EXAMPLE/a1.txt | sed -e "s/\$/+10/g" > a1.txt___ac___10

Calcul Québec

BqTools : continue – 3

/opt/software/bqtools/share/doc/bqtools/bqsubmit.dat.sample

```
batchName=NomChoisi
concurrentJobs=8 # nb max de jobs en concurrence
command=bash monScriptApp.sh # commade/application à exécuter
templatesFiles=a.txt b.txt c.txt #  fichier tempalte
param1=a=$(cat par_a.txt) # les valeurs pour remplacer ~~a~~
param2=b=$(seq 10 -1 1) # nb cas total == produit cartésien
MonConstA=1111 # constant
ParamSymLinks=1 # créer lien symbolique vers repertoire de tâche
submitOptions=" --nodes=1 –time=1:00 –ntasks=2" # options slurm
```

Calcul Québec

BqTools : continue – 4

/opt/software/bqtools/share/doc/bqtools/bqsubmit.dat.sample

copyFiles=acp.txt bcp.txt # fichiers a copier

linkFiles=aln.txt bln.txt # fichiers dont les liens seront crées

microJobs=4 # chaque job soumis exécute 4 tâches parmétrisées

ConcurrentMicroJobs=2 # 2 s'exécutent en même temps, #0=>$SLURM_NTASKS

 # liste des clés et valeurs correspondantes (optionnel)

keysList="varA varB"

ValuesList="::: 1 2 :::+ 3 4" # lien par correspondance

Calcul Québec

BqTools : continue – 5

https://wiki.calculquebec.ca/w/BqTools/en

preBatch = rm -f totalOutput.txt # nœud intéractive, une fois,

#avant soumission des tâches

postBatch= cat *.BQ/*.BQ/output.txt >> totalOutput.txt # assembler résultat

 # nœud intéractive, une fois, après exécution de toutes les tâches

PreJob=… # nœud intéractive, une fois par tâche, dans repertoire de tâche

PostJob=..# nœud intéractive, une fois après chaque tâche,

#dans repertoire de tâche=