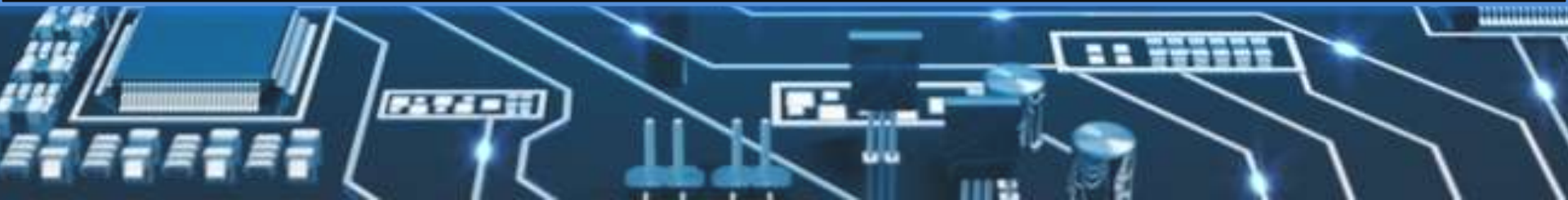
Projet TER L3



UNIVERSITÉ DE MONTPELLIER

LIRMM

Make-cluster

Soutenu par : Thibault ODORICO, Antoine GOUYON,
Jonas FELIX, Romain SIDHOUM

Professeur encadrant : David DELAHAYE

2017/2018

Sommaire

Introduction

1. Mise en contexte et définitions

    1.1. Make et make -j
    1.2. Le cluster

2. Expérimentations et Implémentation

    2.1. Scripts et observations
    2.2. Programme de parallélisation
    2.3. Implémentation

3. Démonstration du programme

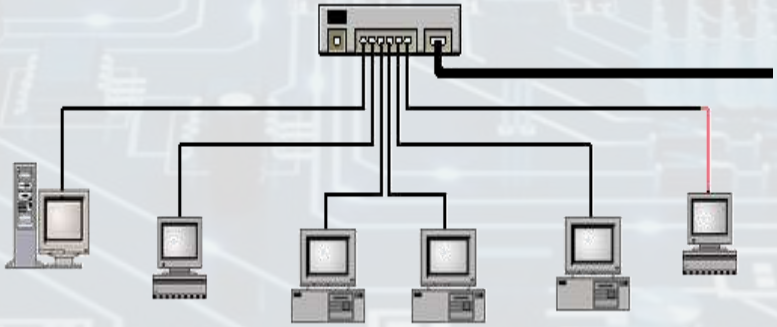
Conclusion

Introduction

- Exécuter des taches rapidement et facilement.

- Deux outils
  → Le make avec l'option –j
  → Le cluster HPC@LR à disposition au LIRMM (56 cœurs)

Problématique : Comment appliquer les calculs du make -j sur le cluster ?



Make et make -j

➢ Make :

- Construction automatique de fichiers exécutables,
- Exécution programmée de commandes bash
- Obtiens un résultat sans nécessairement refaire ses étapes.

➢ Make –j [N]:

- Exécution en parallèle des tâches définies,
- Gain de temps / Rapidité de la commande.
- Exécution sur une machine à la fois

```
1   CC=gcc
2   CFLAGS=-std=c99
3   SRC=fichier1.c fichier2.c fichier3.c ...
4   OBJ=$(SRC:.c=.o)
5
6   all: $(OBJ)
7
8   .SUFFIXES: .c .o
9
10  .c.o:
11      $(CC) -c $(CFLAGS) $<
12
13  clean:
14      rm -f $(OBJ) *~
15
```

Cluster : définition

- Assemblage d'ordinateurs appelés *nœuds,*
- Dépasse la limitation d'un seul ordinateur,
- Augmentation des performances,
- Utilisé dans les calculs haute performance et l'imagerie numérique,
- Gestion des ressources des nœuds autonome.

Cluster : commandes utiles

Commande srun

- Usage: srun [OPTIONS…] executable [args…]

```
[delahayed@muse-login01 make-cluster]$ srun --partition=lirmm7 sleep 3
srun: job 708163 queued and waiting for resources
srun: job 708163 has been allocated resources
```

→ Exécute une commande sur le cluster

Cluster : commandes utiles

Commande squeue

```
[delahayed@muse-login01 make-cluster]$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
            708021    lirmm7 gen_dbch bouvierc PD       0:00      1 (Resources)
            708022    lirmm7 gen_dbch bouvierc PD       0:00      1 (Priority)
            708023    lirmm7 gen_dbch bouvierc PD       0:00      1 (Priority)
            708024    lirmm7 gen_dbch bouvierc PD       0:00      1 (Priority)
            708025    lirmm7 gen_dbch bouvierc PD       0:00      1 (Priority)
            708026    lirmm7 gen_dbch bouvierc PD       0:00      1 (Priority)
            708027    lirmm7 gen_dbch bouvierc PD       0:00      1 (Priority)
            707797    lirmm7 gen_prac bouvierc  R 1-13:41:14      1 muse023
            708019    lirmm7 gen_dbch bouvierc  R    2:43:06      1 muse022
```

Commande sinfo

```
[delahayed@muse-login01 make-cluster]$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
lirmm1       up 1-00:00:00      2    mix muse[022-023]
lirmm7       up 7-00:00:00      2    mix muse[022-023]
```

Scripts et observations

→ Premier script : envoie de la commande sleep 30 fois sur le cluster

```
1   #!/bin/sh
2
3   for i in `seq 30 60`; do
4       srun --partition=lirmm1 sleep $i &
5   done
```

→ Les sleeps sont envoyés dans la file d'attente et exécutés en parallèles

```
708164    lirmm1    sleep delahaye  R       0:23      1 muse022
708165    lirmm1    sleep delahaye  R       0:23      1 muse022
708166    lirmm1    sleep delahaye  R       0:23      1 muse022
708167    lirmm1    sleep delahaye  R       0:23      1 muse022
708168    lirmm1    sleep delahaye  R       0:23      1 muse022
708169    lirmm1    sleep delahaye  R       0:23      1 muse022
708170    lirmm1    sleep delahaye  R       0:23      1 muse022
708171    lirmm1    sleep delahaye  R       0:23      1 muse022
708172    lirmm1    sleep delahaye  R       0:23      1 muse023
708173    lirmm1    sleep delahaye  R       0:23      1 muse023
708174    lirmm1    sleep delahaye  R       0:23      1 muse023
708175    lirmm1    sleep delahaye  R       0:23      1 muse023
708176    lirmm1    sleep delahaye  R       0:23      1 muse023
708177    lirmm1    sleep delahaye  R       0:23      1 muse023
708178    lirmm1    sleep delahaye  R       0:23      1 muse023
708179    lirmm1    sleep delahaye  R       0:23      1 muse023
```

Scripts et observations

→ Deuxième script : envoie de n'importe quel programme avec n'importe quelle donnée au cluster

```
 9  if [[ $# < 2 ]]; then
10      echo "Usage : $0 [EXECUTABLE] [DONNEES]"
11      exit
12  fi
13
14  echo "Running on: $SLURM_NODELIST"
15
16  STARTTIME=$(date +%s.%N)
17
18  for arg do
19      if [[ "$arg" > "1" && "$arg" != "$1" ]]; then
20          srun -n 1 --exclusive $1 $arg &
21      fi
22  done
23
24  wait
25
26  echo "All jobs done !"
27
```

Programme de parallélisation

- La solution srun make –j ?

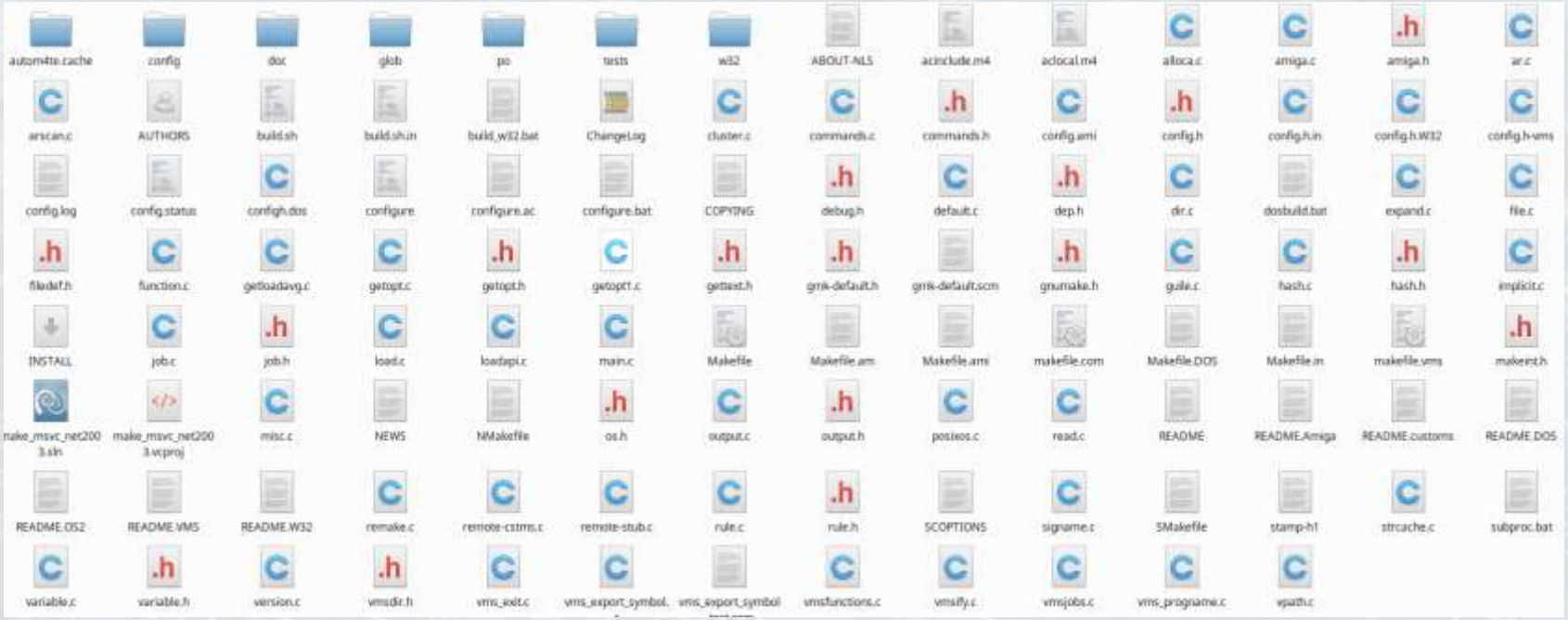
→ Problème

- Modification du code source inévitable

EXPÉRIMENTATIONS ET RÉSULTATS

Implémentation

Code source du gnu make



→ Fichiers intéressants: job.c , main.c

Implémentation

- Identification de la fonction qui exécute les jobs (-j)

```
2020    int
2021    child_execute_job (struct output *out, int good_stdin, char **argv, char **envp)
```

Objectif : exécuter chaque job sur le cluster

Modification du code source

→ *srun make –j compatible*

```
2147    /* PATCH POUR EXECUTER LE JOB VIA SLURM */
2148
2149    char** argx=NULL;
2150
2151    /* Si la variable SLURM_JOB_ID est dans l'environnement, on a accès au cluster */
2152    if(getenv("SLURM_JOB_ID")) {
2153
2154      unsigned int i, argc=4;
2155
2156      /* Compte le nombre d'arguments total */
2157      for(i = 0 ; argv[i] != NULL ; i++)
2158        argc++;
2159
2160      argx = (char**)malloc(sizeof(char*) * argc);
2161
2162      /* Commande d'execution sur le cluster */
2163      argx[0] = "srun";
2164      argx[1] = "-N1";
2165      argx[2] = "-n1";
2166
2167      /* Copie des arguments argv vers argx */
2168      for(i = 0 ; argv[i] != NULL ; i++)
2169        argx[i + 3] = argv[i];
2170
2171      argx[argc - 1] = NULL;
2172
2173      /* Ajout de la commande srun et ces options sur argv */
2174      argv = argx;
2175    }
2176
2177    /* FIN DU PATCH */
```

Implémentation

- Objectif : créer l'option *make –c|--cluster= [OPTIONS,…]*
- Modification du fichier *main.c*
- *Détection des usages dans le code*
- *Parser l'option –c|--cluster*
- *Conserver le fonctionnement du make original*

EXPÉRIMENTATIONS ET RÉSULTATS

Implémentation

```
337 ▼ static const char *const usage[] =
338 ▼   {
339       N_("Options:\n"),
340       N_("\
341     -b, -m                      Ignored for compatibility.\n"),
342       N_("\
343     -B, --always-make           Unconditionally make all targets.\n"),
344       N_("\
345     -C DIRECTORY, --directory=DIRECTORY\n\
346                                 Change to DIRECTORY before doing anything.\n"),
347       N_("\
348     -d                          Print lots of debugging information.\n"),
349       N_("\
350     --debug[=FLAGS]             Print various types of debugging information.\n"),
351       N_("\
352     -e, --environment-overrides\n\
353                                 Environment variables override makefiles.\n"),
354       N_("\
355     --eval=STRING               Evaluate STRING as a makefile statement.\n"),
356       N_("\
357     -f FILE, --file=FILE, --makefile=FILE\n\
358                                 Read FILE as a makefile.\n"),
359       N_("\
360     -h, --help                  Print this message and exit.\n"),
361       N_("\
362     -i, --ignore-errors         Ignore errors from recipes.\n"),
363       N_("\
364     -I DIRECTORY, --include-dir=DIRECTORY\n\
365                                 Search DIRECTORY for included makefiles.\n"),
366       N_("\
367     -j [N], --jobs[=N]          Allow N jobs at once; infinite jobs with no arg.\n"),
368       N_("\
369 ▼   -c [partition=NAME,...], --cluster[=partition=NAME,...]\n\
370                                 Execute jobs on a partition of a cluster.\n\
371                                 Use sinfo to see available partitions.\n\
372                                 Combine with [-j|--jobs] to parallelise jobs.\n"),
```

```
[delahayed@muse-login01 make-cluster]$ make-cluster --help
Usage: make [options] [target] ...
Options:
  -b, -m                      Ignored for compatibility.
  -B, --always-make           Unconditionally make all targets.
  -C DIRECTORY, --directory=DIRECTORY
                              Change to DIRECTORY before doing anything.
  -d                          Print lots of debugging information.
  --debug[=FLAGS]             Print various types of debugging information.
  -e, --environment-overrides
                              Environment variables override makefiles.
  --eval=STRING               Evaluate STRING as a makefile statement.
  -f FILE, --file=FILE, --makefile=FILE
                              Read FILE as a makefile.
  -h, --help                  Print this message and exit.
  -i, --ignore-errors         Ignore errors from recipes.
  -I DIRECTORY, --include-dir=DIRECTORY
                              Search DIRECTORY for included makefiles.
  -j [N], --jobs[=N]          Allow N jobs at once; infinite jobs with no arg.
  -c [partition=NAME,...], --cluster[=partition=NAME,...]
                              Execute jobs on a partition of a cluster.
                              Use sinfo to see available partitions.
                              Combine with [-j|--jobs] to parallelise jobs.
  -k, --keep-going            Keep going when some targets can't be made.
  -l [N], --load-average[=N], --max-load[=N]
                              Don't start multiple jobs unless load is below N.
  -L, --check-symlink-times   Use the latest mtime between symlinks and target.
  -n, --just-print, --dry-run, --recon
                              Don't actually run any recipe; just print them.
  -o FILE, --old-file=FILE, --assume-old=FILE
                              Consider FILE to be very old and don't remake it.
  -O[TYPE], --output-sync[=TYPE]
                              Synchronize output of parallel jobs by TYPE.
  -p, --print-data-base       Print make's internal database.
  -q, --question              Run no recipe; exit status says if up to date.
  -r, --no-builtin-rules      Disable the built-in implicit rules.
  -R, --no-builtin-variables  Disable the built-in variable settings.
  -s, --silent, --quiet       Don't echo recipes.
  -S, --no-keep-going, --stop
                              Turns off -k.
  -t, --touch                 Touch targets instead of remaking them.
  --trace                     Print tracing information.
  -v, --version               Print the version number of make and exit.
  -w, --print-directory       Print the current directory.
  --no-print-directory        Turn off -w, even if it was turned on implicitly.
  -W FILE, --what-if=FILE, --new-file=FILE, --assume-new=FILE
                              Consider FILE to be infinitely new.
  --warn-undefined-variables  Warn when an undefined variable is referenced.

This program built for x86_64-pc-linux-gnu
Report bugs to <bug-make@gnu.org>
```

Implémentation

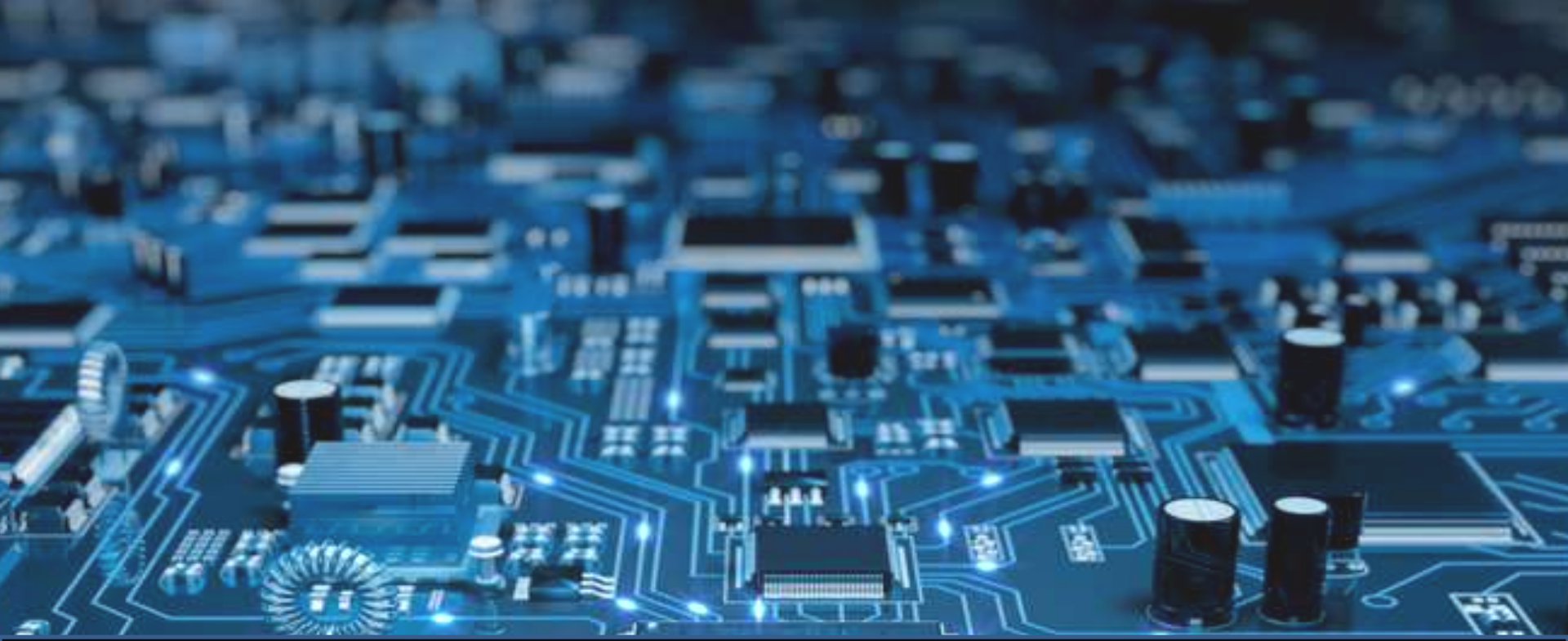
```
1072  main (int argc, char **argv, char **envp)
1073  #endif
1074  {
1075
1076    /* PATCH CLUSTER */
1077    char cluster_opts[1024] = "";
1078
1079    /* Verifie si un option de cluster est presente dans argv et renvoie la chaine d'options si c'est la cas */
1080    if(get_cluster_opt(argc, argv, cluster_opts)){
1081
1082      unsigned int i = 0;
1083
1084      printf("cluster opts: %s\n", cluster_opts);
1085
1086      char make_cluster_command[1024] = "srun ";
1087      char make_options[1024] = "";
1088
1089      char formated_opts[1024] = "";
1090
1091      /* Formate la chaine d'options pour la rendre compatible avec srun */
1092      format_cluster_opts(cluster_opts, formated_opts);
1093
1094      /* Si aucune partition n'est specifiée dans les options du cluster on quitte le programme */
1095      if(!have_partition()){
1096        fprintf(stderr, "%s: [-c|--cluster] no partition precised\n", argv[0]);
1097        return -1;
1098      }
1099
1100      strcat(make_cluster_command, formated_opts);
1101
1102      /* Recupère les options qui ne sont pas liées au cluster et les concatène dans la chaine make_options */
1103      for (i = 0; i < argc; ++i){
1104        if(strcmp(argv[i], "-c") != 0 && strstr(argv[i], "--cluster") == NULL && strcmp(argv[i], cluster_opts) != 0){
1105          strcat(make_options, argv[i]);
1106          strcat(make_options, " ");
1107        }
1108      }
1109
1110      strcat(make_cluster_command, make_options);
1111
1112      /* Enveloppement la commande make avec un srun */
1113      system(make_cluster_command);
1114
1115      return 0;
1116    }
1117    /* FIN PATCH CLUSTER */
```

DÉMONSTRATION

Vidéo démonstrative

Conclusion

- Modification d’un code source inconnu en c
- Difficulté de lecture d’un code avec des exceptions pour plusieurs machines et plusieurs Systèmes d’Exploitations
- Grande arborescence de fichiers, beaucoup de lecture
- Apprentissage et utilisation de nouvelles technologies



Merci de votre attention !