Programmation parallèle :

Message Passing Interface (MPI-1)

Geneviève Moguilny
moguilny@ipgp.jussieu.fr

Institut de Physique du Globe de Paris

Architectures et modèles de programmation

D'après Flynn (1972),
S : Single, **I** : Instruction, **M** : Multiple, **D** : Data.
Séquentielles ou Parallèles

		Contrôle (Inst.)	
		S	I
Données	S	SISD	MISD
	M	SIMD	**MIMD**

Parmi les MIMD, **systèmes répartis** = ensemble de machines reliées par un réseau.

Programmation parallèle ⇒ écriture d'un (ou plusieurs) programme(s) qui s'exécutera(ont) simultanément sur plusieurs processeurs afin d'atteindre un but commun :

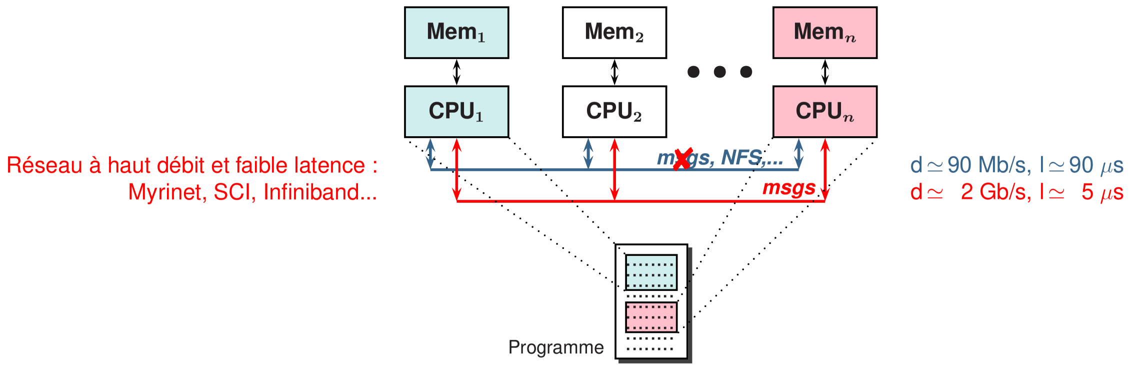
- chaque processeur exécutera une instance de programme,
- échange de données (message) ⇒
  appel à un sous-programme d'une **librairie d'échange (ou de passage) de messages** comme PVM (*Parallel Virtual Machine*) ou MPI (*Message Passing Interface*).

Modèles de programmation sur les systèmes répartis à mémoire distribuée (**P** = Program) :

- MPMD pour un réseau hétérogène ⇒ PVM ou MPI-2,
- **SPMD** pour un réseau homogène ⇒ PVM (obsolète) ou MPI-[1 | 2].

Application SPMD

- Un programme écrit dans un langage classique (Fortran, C...).
- Chaque processus reçoit la même occurence du programme mais par le biais d'instructions conditionnelles, exécute le même code ou une partie différente du code, sur les mêmes données ou des données différentes.
- Les variables du programme sont privées et résident dans la mémoire du processeur allouée au processus.
- Les données sont échangées entre les processus par des appels à des sous-programmes d'une librairie d'échange de messages, comme **MPI**.



La librairie de passage de messages MPI

Historique :

1992 : Nécessité de créer des applications portables avec de bonnes performances ⇒ création d'un groupe de travail (principalement américano-européen) afin d'adopter les méthodes HPF (*High Performance Fortran*).

1994 : Version 1.0 de MPI.

1997 : Définition d'une norme pour MPI-2 (contrôle dynamique des tâches, E/S parallèles...) disponible aujourd'hui.

Principales implémentations du domaine public :

- LAM MPI (*Indiana University*),
- MPICH (*Argonne National Laboratory*) ;

et sur celles-ci,

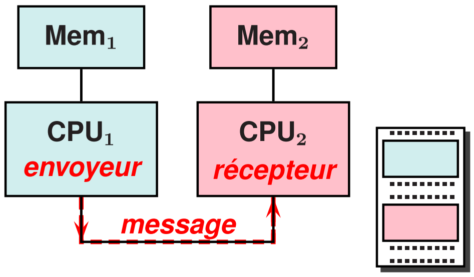
- librairies spécifiques à un réseau (MPI-GM : implémentation de MPICH sur Myrinet).

Concept d'échange de message

Application MPI = ensemble de processus autonomes exécutant leur propre code, et communiquant via des appels à des sous-programmes de la bibliothèque MPI.

Communication ⇒ Échange de messages

Message = identificateur(processus émetteur) + type des données (simple ou dérivé) + longueur + identificateur(processus récepteur) + **donnée**



Communication point à point

Principales catégories de l'API MPI

❶ Gestion de l'environnement

- activation / désactivation de l'environnement (**MPI_INIT**, **MPI_FINALIZE**),
- identification des processus (**MPI_COMM_SIZE**, **MPI_COMM_RANK**).

❷ **Communications** (envoi et réception de messages)

- **point à point**,
- **collectives**.

❸ **Gestion des groupes et communicateurs**

Communicateur : ensemble de processus sur lesquels portent les opérations MPI, groupe de processeurs + contexte de communication (géré par MPI).

Création d'un communicateur à partir :

- d'un groupe de processus préalablement crée par le programmeur,
- d'un autre communicateur (**MPI_COMM_WORLD** = l'ensemble des processus).

Tout petit exemple

- **Source : HelloMPI.f90**

```fortran
program HelloMPI

implicit none
include 'mpif.h'
integer :: nb_procs, rang, ierr

call MPI_INIT(ierr)

call MPI_COMM_SIZE(MPI_COMM_WORLD, nb_procs, ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, rang, ierr)
print *, 'Je suis le processus ', rang,' parmi ',nb_procs

call MPI_FINALIZE(ierr)

end program HelloMPI
```


Tout petit exemple (suite)


```fortran
program HelloMPI

implicit none
include 'mpif.h'
integer :: nb_procs, rang, ierr

call MPI_INIT(ierr)

call MPI_COMM_SIZE(MPI_COMM_WORLD, nb_procs, ierr)
call MPI_COMM_RANK(MPI_COMM_WORLD, rang, ierr)
print *, 'Je suis le processus ', rang,' parmi ',nb_procs

call MPI_FINALIZE(ierr)

end program HelloMPI
```


- **Compilation et Édition de liens :**

`mpif90 HelloMPI.f90 -o HelloMPI`

⇕

`ifort -c -I`*includes_path* **`HelloMPI.f90`**
`ifort -L`*libs_path* **`HelloMPI.o -static-libcxa -o HelloMPI -lmpichf90 -lmpich`**

- **Lancement de l'exécution :**

`mpirun -np 4 -machinefile mf HelloMPI`

Lancement de l'exécution de **`HelloMPI`** (par `ssh` ou `rsh`) sur 4 processeurs (**`np`** = *number of processors*) des machines parmi celles citées dans le fichier **`mf`**.

- **Résultat obtenu :**

```
Je suis le processus          0  parmi          4
Je suis le processus          3  parmi          4
Je suis le processus          1  parmi          4
Je suis le processus          2  parmi          4
```

Les communications point à point

Communications entre 2 processus identifiés par leur rang dans le communicateur.

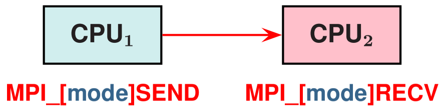
Modes de transfert :

standard :	MPI effectue ou non une recopie temporaire suivant l'implémentation.
***S**ynchronous :*	envoi terminé quand réception achevée.
***B**uffered :*	recopie temporaire dans un buffer crée par le programmeur, envoi terminé lorsque recopie achevée.
***R**eady :*	envoi effectué lorsque réception initiée (clients/serveurs).

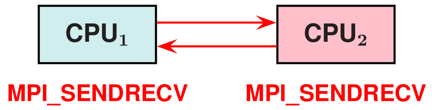
Envoi bloquant (envoi et réception couplés) ou non, réception **I**mmédiate ou non.

Deux types d'échange :

1. 1 envoyeur et 1 récepteur distincts (MPI_SEND, MPI_SSEND, MPI_IBSEND ...) :

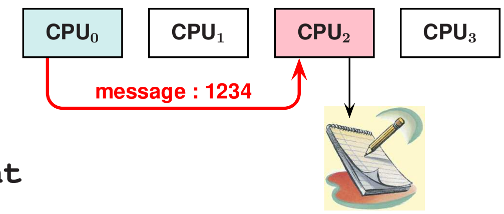


2. l'envoyeur reçoit aussi et inversement (MPI_SENDRECV, MPI_SENDRECV_REPLACE) :



Exemple de communication point à point simple

```
!! point_a_point.f90 : Exemple d'utilisation de MPI_SEND et MPI_RECV
!! Auteur : Denis GIROU (CNRS/IDRIS - France) <Denis.Girou@idris.fr> (1996)
program point_a_point
  implicit none
  include 'mpif.h'
  integer, dimension(MPI_STATUS_SIZE) :: statut
  integer, parameter                  :: etiquette=100
  integer                             :: rang,valeur,ierr
  call MPI_INIT(ierr)
  call MPI_COMM_RANK(MPI_COMM_WORLD,rang,ierr)

  if (rang == 0) then
     valeur=1234
     call MPI_SEND(valeur,1,MPI_INTEGER,2,etiquette,MPI_COMM_WORLD,ierr)
  elseif (rang == 2) then
     call MPI_RECV(valeur,1,MPI_INTEGER,0,etiquette,MPI_COMM_WORLD,statut,ierr)
     print *,'Moi, processus ', rang, ' j''ai reçu ',valeur,' du processus 0.'
  end if

  call MPI_FINALIZE(ierr)
end program point_a_point
```



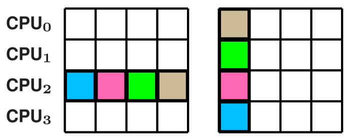

```
mpif90 point_a_point.f90 -o point_a_point

mpirun -np 4 -machinefile mf point_a_point
```

```
Moi, processus           2  j'ai reçu         1234  du processus 0.
```

Les communications collectives

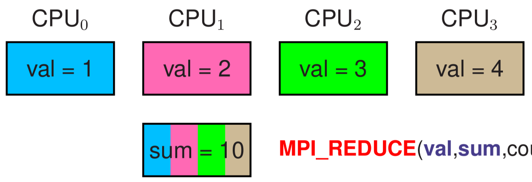
<u>Ensemble</u> de communications point à point à l'intérieur d'un groupe, fait en <u>une</u> opération.

- Synchronisations (BARRIER)
- Mouvement de données
  - Diffusions générales (BCAST) ou sélectives (SCATTER)



MPI_SCATTER(valeurs,ltranche,MPI_REAL, &
    donnees,ltranche,MPI_REAL,2,MPI_COMM_WORLD,code)

  - Collectes réparties (GATHER) ou générales (ALLGATHER)
  - Échanges croisés (ALLTOALL)
- Réductions (REDUCE avec SUM, PROD, MAX, MIN, MAXLOC, MINLOC, IAND, IOR, IXOR)



MPI_REDUCE(**val**,**sum**,count,MPI_INTEGER,**MPI_SUM**,0,MPI_COMM_WORLD,code)

Création de communicateurs : les topologies

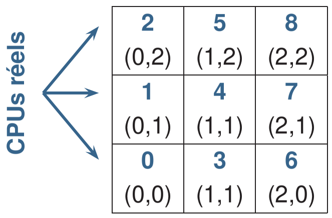
Pour les applications qui utilisent une décomposition de domaine,
il est intéressant de définir une grille virtuelle des processus afin d'en faciliter la manipulation :

nombre de sous-domaines = nombre de processus,
travail sur les sous-domaines puis finalisation par calculs aux frontières.

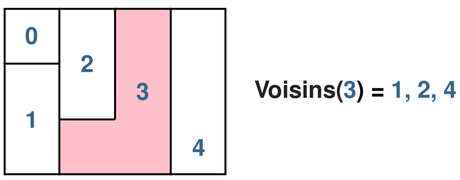
$\Rightarrow$ création de nouveaux **communicateurs** sur ces nouvelles topologies.

Ces topologies peuvent être :

- **cartésiennes** : grille périodique ou non, identification des processus par leurs coordonnées,
- de type **graphe** pour les topologies plus complexes.



MPI_CART_CREATE(...)



MPI_GRAPH_CREATE(...)

Développement d'une application MPI

Pourquoi MPI ?

- Permet de changer l'ordre de grandeur du problème à traiter par une augmentation possible quasi-illimitée de la mémoire disponible, avec du matériel et du logiciel de base.

Optimisation

- Choix du mode de communication : éviter les recopies inutiles
- Contention des messages
- Équilibrage de la charge en faisant attention aux goulets d'étranglement (E/S)
- Recouvrement des temps de calcul et de communications
- Utilisation des communications persistantes

Aide au développement

- Débogueurs graphiques supportant MPI : TotalView, DDT
- Visualisation des communications : Vampir/VampirTrace, Blade

Application bien parallélisée avec MPI ⇒
temps d'exécution inversement proportionnel au nombre de processeurs utilisés

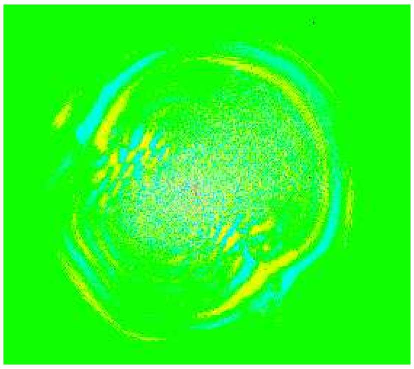
Documentation MPI

- Programmation parallèle et répartie :
  `http://www.gulus.org/spip/IMG/pdf/midi_linux_mpi.pdf`
- Cours MPI :
  `http://www.idris.fr/data/cours/parallel/mpi/choix_doc.html`
  `www.obspm.fr/~websio/documentations/parallel/archi_mpi3.pdf`
  `http://www.cines.fr/textes/support_para.html`
- Implémentations MPI du domaine public :
  - LAM : `http://www.lam-mpi.org`
  - MPICH : `http://www.mcs.anl.gov/mpi/mpich`

Exemple d'application MPI tournant sur EGEE : SPECFEM3D

Résolution de problèmes de propagation d'ondes sismiques dans des milieux complexes et à haute fréquence à l'échelle régionale, par la méthode des éléments spectraux.

Application écrite initialement par D. Komatitsch (Université de Pau), utilisée dans plus de 75 laboratoires dans le monde, essentiellement pour des études de tremblements de terre, tomographie sismique, ultrasons dans les cristaux, topographie, bassins et vallées sédimentaires, ondes d'interface, et cristaux anisotropes.



Informations supplémentaires : `http://www.univ-pau.fr/~dkomati1`

Positionnement **EGEE** : NA4, VO ESR, catégorie *Physique de la Terre solide*

SPECFEM3D : spécificités techniques

- $\simeq$ 20 000 lignes de Fortran 90 / MPI.
- Peut tourner sur 4 CPUs, a tourné sur 1944 CPUs sur le *EarthSimulator* (Japon).
  Sur EGEE, a tourné sur 64 CPUs au Nikhef (Pays-Bas), et sur 16 CPUs à SCAI (Allemagne), au LAL, au CPPM, et à la CGG.
- **Contraintes :**
  - Optimisation de la mémoire $\Longrightarrow$ recompilation et mise à jour de fichiers d'entrée sur SE nécessaires à chaque changement des paramètres d'entrée.
  - Beaucoup de sorties provisoires ($\Longrightarrow$ écriture dans les `/tmp` propres à chaque nœud), + sorties écrites dans un répertoire partagé à récupérer **($\Longrightarrow$ `/home` montés NFS)**.
  - Nécessite le lancement successif de **deux mpirun** qui doivent utiliser les mêmes nœuds, alloués dans le même ordre.

SPECFEM3D sur EGEE : Fem4.jdl

```
#   Fem4.jdl
    NodeNumber = 4;
    VirtualOrganisation = "esr";
    Executable = "Fem";
    Arguments = "xmeshfem3D4 xspecfem3D4 4";
    StdOutput = "Fem.out";
    StdError = "Fem.err";
    JobType = "mpich";
    LRMS_type = "pbs";
    Type = "Job";
    InputSandbox = { "Fem", "xmeshfem3D4", "xspecfem3D4"};
    OutputSandbox = { "output_mesher.txt","output_solver.txt", "xmeshfem3D4.out",
                      "xspecfem3D4.out", "Fem.out","Fem.err","Par_file"};
#    Requirements = (other.GlueCEInfoTotalCPUs >= 4);

```


SPECFEM3D sur EGEE : script Fem (1/2)

```
#!/bin/sh
# copy of scai:/home/lcg2/mpirun_g95 modified - 11/03/2005
#
EXE1=$1
EXE2=$2
CPU_NEEDED=$3
TMP_DIR=/tmp/DATABASES_MPI_DIMITRI4
institut=`echo $HOSTNAME | cut -f2-5 -d.`
# Different path for mpirun on LAL and CGG
if  [ $institut == "lal.in2p3.fr" -o $institut == "private.egee.fr.cgg" ] ; then
   MPIHOME=/usr/bin
else
   MPIHOME=/opt/mpich/bin
fi
if [ -f "$PWD/.BrokerInfo" ] ; then
TEST_LSF=`edg-brokerinfo getCE | cut -d/ -f2 | grep lsf`
else
TEST_LSF=`ps -ef | grep sbatchd | grep -v grep`
fi
if [ "x$TEST_LSF" = "x" ] ; then
   HOST_NODEFILE=$PBS_NODEFILE
else
   echo "LSF Hosts: $LSB_HOSTS"
   HOST_NODEFILE=`pwd`/lsf_nodefile.$$
fi
for host in `cat ${HOST_NODEFILE}`
do
  ssh $host rm -rf $TMP_DIR
  ssh $host mkdir $TMP_DIR
  ssh $host chmod 775 $TMP_DIR
done
LocalFn="DATA.tgz"
echo "trying edg-rm..."
cmd="edg-rm --vo esr copyFile lfn:DATA${CPU_NEEDED}.tgz file://`pwd`/$LocalFn"
echo ">>> $cmd"
$cmd

if [ ! -f $LocalFn ] ; then
   echo "edg-rm failed, trying lcg-cp..."
   export  LCG_GFAL_INFOSYS="mu3.matrix.sara.nl:2170"
   cmd="lcg-cp --vo esr lfn:DATA${CPU_NEEDED}.tgz file://`pwd`/$LocalFn"
   echo ">>> $cmd"
   $cmd
fi
```

*Définition du path de **mpirun***

Création des répertoires temporaires

Rapatriement des données d'un SE

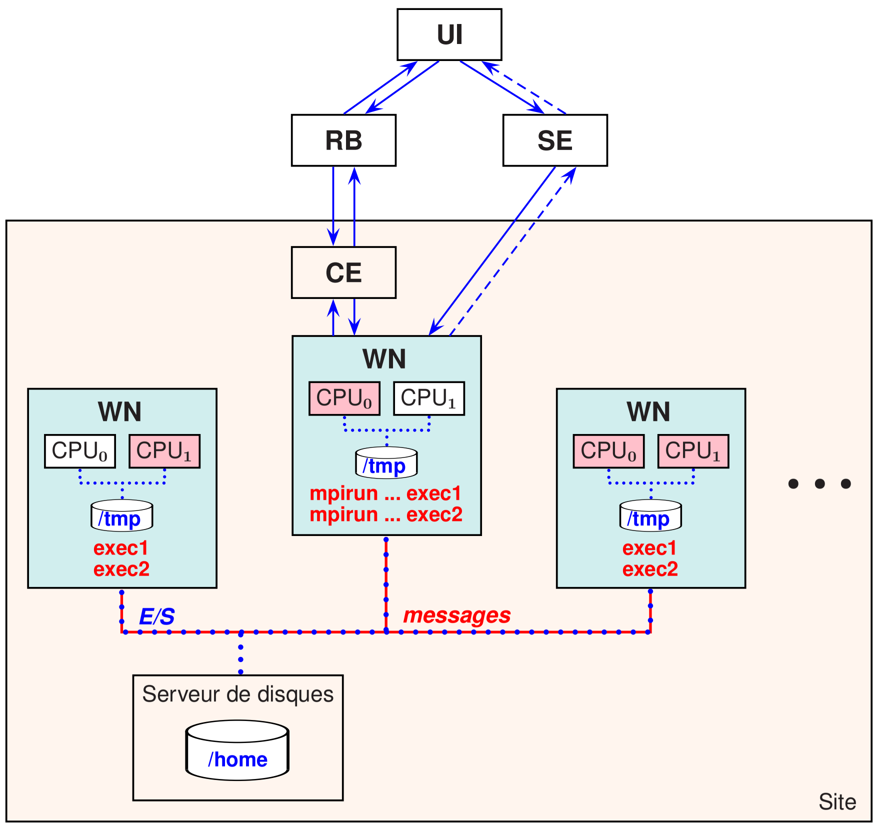
SPECFEM3D sur EGEE : script Fem (2/2)

```
if [ ! -f $LocalFn ] ; then
   echo "$LocalFn not found."
   exit
fi
#
echo "**************************************"
tar xfz $LocalFn
rm $LocalFn
rm -rf OUTPUT_FILES
mkdir OUTPUT_FILES
cp DATA/Par_file .

chmod 755 $EXE1
chmod 755 $EXE2
ls -l
time $MPIHOME/mpirun -np $CPU_NEEDED -machinefile $HOST_NODEFILE \
      `pwd`/$EXE1 > $EXE1.out
cp OUTPUT_FILES/output_mesher.txt .
time $MPIHOME/mpirun -np $CPU_NEEDED -machinefile $HOST_NODEFILE \
      `pwd`/$EXE2 > $EXE2.out
echo "Size of OUTPUT_FILES :"
du -sk OUTPUT_FILES
for host in `cat ${HOST_NODEFILE}`
do
  ssh $host echo "Size of $TMP_DIR on $host :"
  ssh $host du -sk $TMP_DIR
  ssh $host rm -rf $TMP_DIR
done
cp OUTPUT_FILES/output_solver.txt .
tar cfz OUTPUT_FILES${CPU_NEEDED}.tgz OUTPUT_FILES
lcg-del --vo esr -s grid11.lal.in2p3.fr `lcg-lg --vo esr \
        lfn:OUTPUT_FILES${CPU_NEEDED}.tgz`
lcg-cr  --vo esr -d grid11.lal.in2p3.fr \
        file://`pwd`/OUTPUT_FILES${CPU_NEEDED}.tgz \
        -l OUTPUT_FILES${CPU_NEEDED}.tgz
exit
```

*Lancement des **mpirun***

Stockage des sorties sur SE

SPECFEM3D sur EGEE : exécution



Perspectives

- SPECFEM3D : des collaborations avec DEISA et Grid5000 vont être gérées par l'INRIA Bordeaux.
- Souhaits à court terme :
  - Installations homogènes de MPI (*path* standard ou tag MPIHOME).
  - Nœud $\neq$ CPU
    $\Rightarrow$ remplacement de Torque par un autre LRMS comme **SGE** ou **OAR** ?
  - Queues spécifiques MPI contenant des CPUs de puissance équivalente.
- Dans le futur...
  - Réservation de CPUs.
  - MPI inter-sites (**MPICH-G2**, **MPICH-V**).
  - Disposer de réseaux performants (Myrinet, Infiniband...).
  - MPI sur des RC disposant de plusieurs centaines de nœuds (CCIN2P3 ?).
- Mais globalement, **MPI / Fortran90 sur EGEE marche ! !**