Le parallélisme pour ~~les nuls~~ tous

Plan

- Contexte
  - Pour quel public ?
  - Etat des lieux
  - Objectifs
- La méthode
  - Codes cibles
  - Gagner du temps
  - MPI vs OpenMP
  - En pratique 1
  - En pratique 2
- Quelques exemples
  - Résonance électromagnétique : méthode de Monte Carlo
  - EMRODE : biostatistiques
  - EMRODE : OpenMP
  - EMRODE : MPI
  - EMRODE : résultats
- Conclusion
  - Précautions à prendre
  - Aller plus loin

Pour quel public ?

- Contexte du Mésocentre de Calcul Intensif Aquitain (MCIA) : laboratoires de recherche de disciplines variées
- Certaines communautés sans connaissance poussée du calcul intensif, mais disposant de codes de simulation
- Utilisateurs orphelins n'ayant pas investi dans la réduction de la durée des jobs

Etat des lieux

- Certains chercheurs seraient intéressés par le calcul parallèle mais le trouvent trop compliqué
- Une formation à la programmation parallèle ne suffit pas à créer une dynamique
- Faire une parallélisation entière d'un code peut-être long
  - Elle ne sera pas forcément accepté par l'équipe de développement principale
  - La compétence n'est pas transmise aux gens qui maintiennent le code
- A plus long terme, il s'agit d'élargir le champ de recherche de ces utilisateurs

Objectifs

- Efficacité
  - Démontrer les apports d'une approche parallèle
- Intervention « rapide »
  - Pour convaincre
  - Pour pouvoir assurer d'autres activités
- Adoption des modifications proposées
  - Méthode « peu » intrusive pour ne pas brusquer le développeur principal
  - Suffisamment simple pour donner envie de s'approprier les techniques mises en œuvre
- Donner des garanties de validité
  - Exactitude des résultats (modulo la précision machine)
  - Dans un contexte où il peut être compliqué d'obtenir des cas tests, la parallélisation du code peut être incomplète

Plan

- Contexte
  - Pour quel public ?
  - Etat des lieux
  - Objectifs
- La méthode
  - Codes cibles
  - Gagner du temps
  - MPI vs OpenMP
  - En pratique 1
  - En pratique 2
- Quelques exemples
  - Résonance électromagnétique : méthode de Monte Carlo
  - EMRODE : biostatistiques
  - EMRODE : OpenMP
  - EMRODE : MPI
  - EMRODE : résultats
- Conclusion
  - Précautions à prendre
  - Aller plus loin

Codes cibles

- Longues simulations
- Données indépendantes
- Exemples :
  - Méthodes de type Monte Carlo
  - Statistiques
  - Etc...
- Bref : des cas simples !

Gagner du temps

- On se concentre sur la répartition des calculs (distribution de boucles...)
- On néglige une éventuelle répartition des données (lorsque c'est possible)
- Les schémas de communications seront volontairement triviaux, mais :
  - Relativement discrets dans le code
  - Simples à mettre en œuvre
  - Faciles à copier et adapter à d'autres codes similaires
- Des raffinements peuvent être envisagés dans des phases ultérieures de collaboration

MPI vs OpenMP

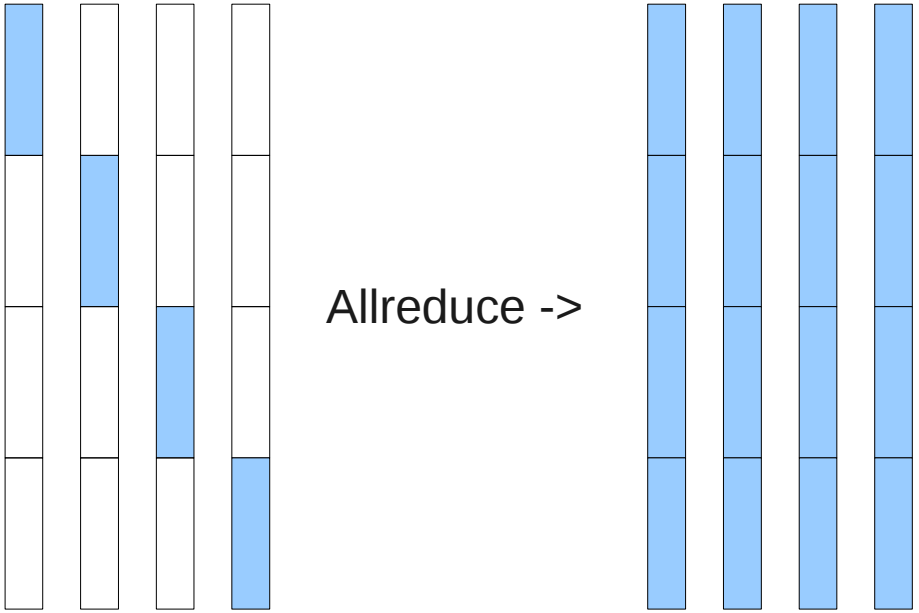
- Comment choisir ?
  - On veut implanter un parallélisme à grains les plus gros possibles
  - On est obligé de travailler sur un code du monde « réel » (ie : pas toujours très beau)
  - On veut que les développeurs principaux adoptent les modifications proposées
  - On veut proposer un spectre d'application le plus large possible
- => MPI (voir comparaison plus loin !)
- Souvent, le client a une idée de ce qu'il veut (« OpenMP matters »)
  - Je pense que c'est un fausse bonne idée

En pratique 1

- Phase préliminaire : on analyse le code :
  - Profiling tout simple sur des jeux de données « significatifs »
  - On en profite pour réclamer des jeux de résultats pour comparaison ultérieure
- Phase 1 : le code doit pouvoir s'exécuter en parallèle, tout en ne parallélisant rien
  - Ajouter `MPI_Init` / `MPI_Finalize`
  - On ne touche pas les lectures de fichiers de données : tous les processeurs lisent en même temps
  - Nettoyer les sorties écran / fichiers
  - Tous les calculs sont effectués par chaque processeur
- Résultat : on fait en parallèle ce qu'on pourrait faire en séquentiel !

En pratique 2

- Phase 2
  - On sélectionne la ou les quelques boucles les plus consommatrices en CPU
  - Distribution des itérations sur les données
  - Reconstitution du résultat global avec `MPI_Allreduce`
  - Chaque processeur peut continuer son travail comme s'il était seul
- Je vous avais prévenu que ça ne volerait pas haut !
- C'est simpliste, mais on a quelque chose qui tourne en parallèle.
- Parfois, on gagne même en temps de calcul !



Plan

- Contexte
  - Pour quel public ?
  - Etat des lieux
  - Objectifs
- La méthode
  - Codes cibles
  - Gagner du temps
  - MPI vs OpenMP
  - En pratique 1
  - En pratique 2
- Quelques exemples
  - Résonance électromagnétique : méthode de Monte Carlo
  - EMRODE : biostatistiques
  - EMRODE : OpenMP
  - EMRODE : MPI
  - EMRODE : résultats
- Conclusion
  - Précautions à prendre
  - Aller plus loin

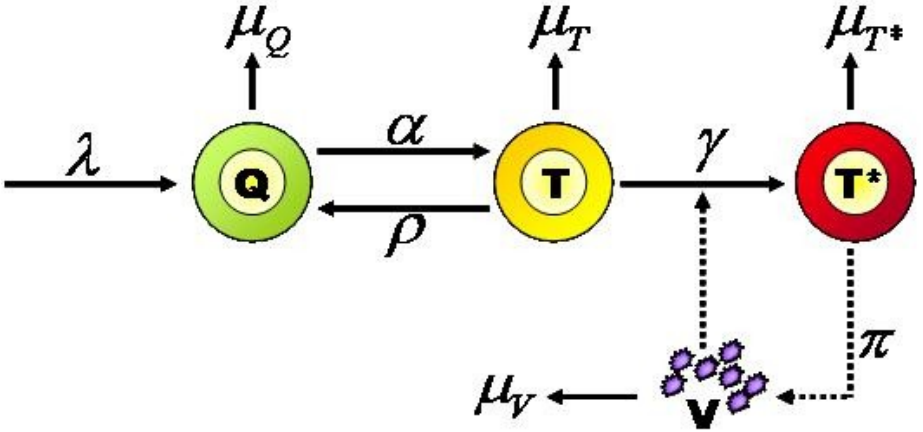
Résonance électromagnétique : méthode de Monte Carlo

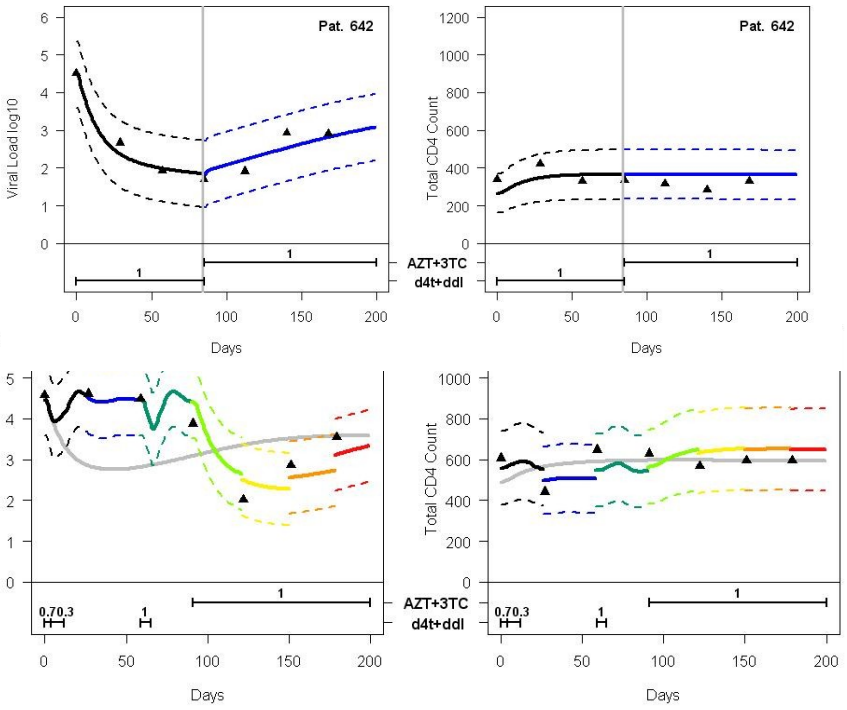
- J. Kliava (LOMA, U. Bx 1)
- Maquette de code fournie par le chercheur
- Premier essai en OpenMP
  - On doit passer en revue tous les commons
  - Au bout de la cinquantième variable globale, on se dit qu'il y a un autre moyen
- Deuxième essai MPI :
  - 95 lignes modifiées sur 1800
  - Seul détail : le générateur pseudo aléatoire
  - Bonnes performances tant que le nombre de processeurs est raisonnablement bas
  - Le chercheur a pu s'approprier la méthode pour la mettre lui même dans son code de production

```
      do 11 Npts=1,Ncent

        if (mod (Npts, mpi_sz) .ne. mpi_rk) then

          tmpalea = alea(1) ! tirage pseudo aleatoire de remplacement
          tmpalea = alea(1) ! tirage pseudo aleatoire de remplacement
          CALL DISTRIB(.false.)

          !cycle
          goto 11
        end if

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! CORPS DE LA BOUCLE
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


  11  continue



      CALL MPI_REDUCE (Spabs(Ninfc:Nsupc), Abs_tmp(Ninfc:Nsupc),
     &     Nsupc-Ninfc+1, MPI_DOUBLE_PRECISION, MPI_SUM,
     &     0, MPI_COMM_WORLD, ierror)
      Spabs(Ninfc:Nsupc) = Abs_tmp(Ninfc:Nsupc)

      CALL MPI_REDUCE (Abs_nc(Ninfc:Nsupc), Abs_tmp(Ninfc:Nsupc),
     &     Nsupc-Ninfc+1, MPI_DOUBLE_PRECISION, MPI_SUM,
     &     0, MPI_COMM_WORLD, ierror)
      Abs_nc(Ninfc:Nsupc) = Abs_tmp(Ninfc:Nsupc)

      do j = 1, 15
         CALL MPI_REDUCE (Abs_sel(Ninfc:Nsupc, j),
     &        Abs_tmp(Ninfc:Nsupc), Nsupc-Ninfc+1,
     &        MPI_DOUBLE_PRECISION, MPI_SUM,
     &        0, MPI_COMM_WORLD, ierror)
        Abs_sel(Ninfc:Nsupc, j) = Abs_tmp(Ninfc:Nsupc)
      end do
```

EMRODE : biostatistiques

- M. Prague, D. Commenges (ISPED)
- Estimation par Maximum a Posteriori de paramètres d'équations différentielles
  - Application à la prédiction de réactions a un changement de traitement chez les patients infectés par le VIH.
- Problème :
  - Petit nombre de calculs indépendant longs (plusieurs heures) faisant partie du processus de recherche
  - Passage en quadruple précision : ~ 20 jours de calcul
- Solution :
  - MPI
  - Oui, mais il vaut quand même mieux faire de l'OpenMP...





EMRODE : OpenMP

- Version M. Prague
- Très efficace jusqu'à 8 cœurs
- Bon équilibrage de la charge
- Quelles perspectives de survie lors du cycle de développement ?

```
C$OMP PARALLEL DEFAULT(PRIVATE)
C$OMP1 FIRSTPRIVATE(b,nql,ier,det,ep,seuil,npm2)
C$OMP2 SHARED(extrema,scalesauv,scaleinvsauv,scaleinv2sauv,nbpatient)
C$OMP3 SHARED(detersauv,startsauv,startsauvind,tmin,tmax,pas2,lambda0)
C$OMP4 SHARED(r0,a0,muq0,gamma0,mut0,muinf0,pi0,muv0,alpha0,eta0,tps2)
C$OMP5 SHARED(tps3,p0,aa0,mup0,rho0,pi10,vrais_obs,npm,schedule)
C$OMP6 SHARED(donnees,nbobs,rltest)
C$OMP7 SHARED(detection,boost,idpat,var_expl,dose,trait1,trait2)
C$OMP8 SHARED(trait3,var_expl2,esp_prior,std_prior)
!$omp9 COPYIN(lambda,r,a,muq,gamma,mut,muinf,pi,muv,alpha,eta)
!$omp1 COPYIN(p,aa,mup,pi1,rho,dpi,dmuq,dgammadb1,dlambda)
!$omp2 COPYIN(dmuinfdb1,b0,b1,prmbio1,adaptive,adaptive2)
!$omp3 COPYIN(numpat1,numcoeff1,ni2,parmalea,simul0,abserr)
!$omp4 COPYIN(abserr1,abserrfuncpa,derivepatbypat,seuil2)
!$omp5 COPYIN(score_cond,befalea,varalea,ndimalea,compteur2)
!$omp6 COPYIN(systeme,rangOMP,nb_tache,numeroalea,recherche)
!$OMP7 COPYIN(fth,fthn,score,recap,test_funcpa,test_funcpa2)
!$OMP7 COPYIN(funcpa_compare,recherche2)
!$OMP8 REDUCTION(+:funcpa)
!$         rangOMP=OMP_GET_THREAD_NUM()
!$         nb_tache=OMP_GET_NUM_THREADS()

C$OMP DO SCHEDULE(RUNTIME)
      DO i=1,nbpatient

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! CORPS DE LA BOUCLE PRINCIPALE
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!


      END DO
!$OMP END DO
!$OMP END PARALLEL
```

EMRODE : MPI

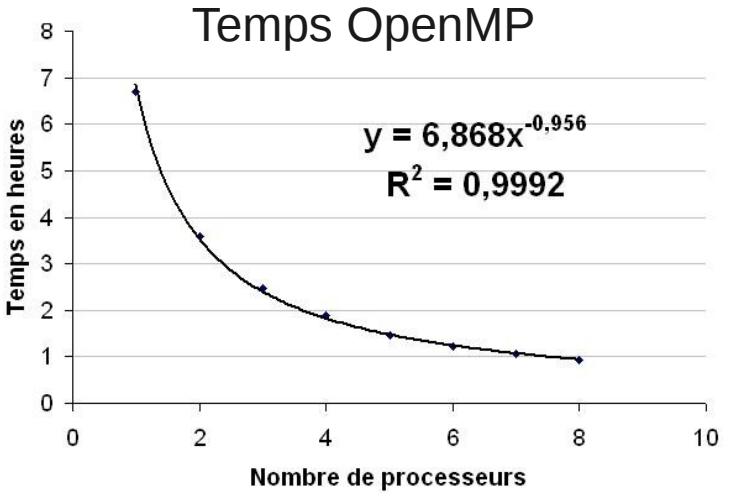
- Version brute du code
- On pourrait dissimuler la complexité : subroutines
- Meilleure espérance de survie

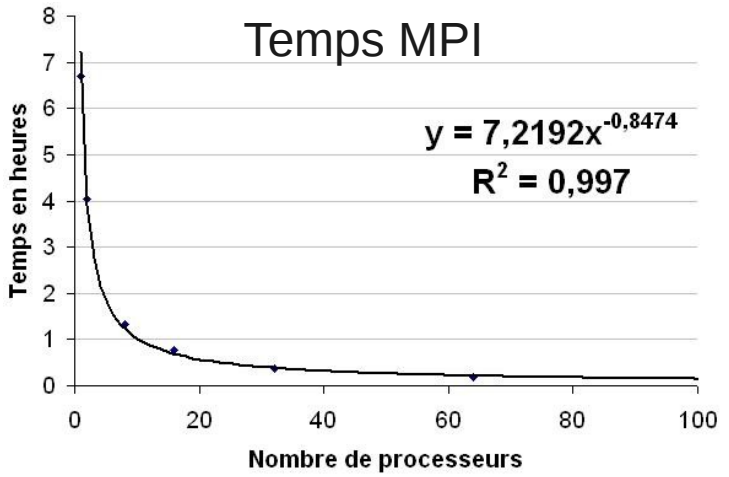
```
      globbounds(1,1) = 1
      globbounds(2,1) = nbpatient
      call distribute_loop (1, globbounds, locbounds)

      rltest=0.d0

! nettoyage des tableaux //
      do i=1,locbounds(1,1)-1
         startsauv(:,i) = 0.D0
         startsauvind(:,i) = 0.D0
         scalesauv(:,:,i) = 0.D0
         scaleinvsauv(:,:,i) = 0.D0
         scaleinv2sauv(:,:,i) = 0.D0
         detersauv(i) = 0.D0
      end do

      do i=locbounds(2,1)+1, nbpatient
         startsauv(:,i) = 0.D0
         startsauvind(:,i) = 0.D0
         scalesauv(:,:,i) = 0.D0
         scaleinvsauv(:,:,i) = 0.D0
         scaleinv2sauv(:,:,i) = 0.D0
         detersauv(i) = 0.D0
      end do

      vrais_obs(:) = 0.D0

      DO i=locbounds(1,1), locbounds(2,1)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! CORPS DE LA BOUCLE PRINCIPALE
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

      END DO
```

```
      if (nbprocs .gt. 1) then
         call MPI_ALLREDUCE (funcpa, reduce0, 1,
     &                       MPI_DOUBLE_PRECISION, MPI_SUM,
     &                       MPI_COMM_WORLD, mpierr)
         funcpa = reduce0

         call MPI_ALLREDUCE (vrais_obs, reduce1, nbpatient,
     &                       MPI_DOUBLE_PRECISION, MPI_SUM,
     &                       MPI_COMM_WORLD, mpierr)
         vrais_obs(1:nbpatient) = reduce1

         call MPI_ALLREDUCE (detersauv, reduce1, nbpatient,
     &                       MPI_DOUBLE_PRECISION, MPI_SUM,
     &                       MPI_COMM_WORLD, mpierr)
         detersauv(1:nbpatient) = reduce1

           call MPI_ALLREDUCE (startsauv, reduce2, ndim*nbpatient,
     &                       MPI_DOUBLE_PRECISION, MPI_SUM,
     &                       MPI_COMM_WORLD, mpierr)
         startsauv(:,1:nbpatient) = reduce2

           call MPI_ALLREDUCE (startsauvind, reduce2, ndim*nbpatient,
     &                       MPI_DOUBLE_PRECISION, MPI_SUM,
     &                       MPI_COMM_WORLD, mpierr)
         startsauvind(:,1:nbpatient) = reduce2

         call MPI_ALLREDUCE (scalesauv, reduce3, ndim*ndim*nbpatient,
     &                       MPI_DOUBLE_PRECISION, MPI_SUM,
     &                       MPI_COMM_WORLD, mpierr)
         scalesauv(:,:,1:nbpatient) = reduce3

         call MPI_ALLREDUCE (scaleinvsauv, reduce3,
     &                       ndim*ndim*nbpatient,
     &                       MPI_DOUBLE_PRECISION, MPI_SUM,
     &                       MPI_COMM_WORLD, mpierr)
         scaleinvsauv(:,:,1:nbpatient) = reduce3

         call MPI_ALLREDUCE (scaleinv2sauv, reduce3,
     &                       ndim*ndim*nbpatient,
     &                       MPI_DOUBLE_PRECISION, MPI_SUM,
     &                       MPI_COMM_WORLD, mpierr)
         scaleinv2sauv(:,:,1:nbpatient) = reduce3
      end if


      funcpa = 0.d0
      do i=1, nbpatient
         result = vrais_obs(i)
         if (result .le. 0.d0) then
            funcpa=-1.d10
            exit
         else

            funcpa=funcpa+log(result)
            recap(i)=log(result)

            if( test_funcpa .and.
     &         funcpa .le. -abs(funcpa_compare)) then

             funcpa=(funcpa*nbpatient)/i
             exit
            end if
         end if

         if (isnan(funcpa) ) then
            funcpa=-1.d10
            exit
         end if
      end do
```

EMRODE : résultats

- En OpenMP, bons résultats jusqu'à 8 cœurs
- En MPI, efficacité 50% pour 100 cœurs
- Equilibrage de charge envisageable mais pas désiré
- Possibilité de combiner les deux approches pour plus d'efficacité
- La méthode est transposable sur beaucoup des codes du laboratoire
- D'autres codes sont portés sur la grille (séquentiel)





Plan

- Contexte
  - Pour quel public ?
  - Etat des lieux
  - Objectifs
- La méthode
  - Codes cibles
  - Gagner du temps
  - MPI vs OpenMP
  - En pratique 1
  - En pratique 2
- Quelques exemples
  - Résonance électromagnétique : méthode de Monte Carlo
  - EMRODE : biostatistiques
  - EMRODE : OpenMP
  - EMRODE : MPI
  - EMRODE : résultats
- Conclusion
  - Précautions à prendre
  - Aller plus loin

Précautions à prendre

- Il faut être intransigeant sur la précision des résultats obtenus en parallèle
  - L'idéal étant d'être au niveau de la précision machine
  - Il faut justifier clairement tout écart supérieur
- Système de gestion de versions
  - A utiliser pour soi-même si l'équipe de chercheurs est réfractaire
  - Sinon, développer de préférence dans le « trunk » pour une meilleure intégration avec les évolutions « mainstream »
- Insister pour obtenir des cas tests
  - Valider les résultats de référence sur la version officielle
  - Bien signaler que les parties du code non couvertes par les cas tests fournis ne seront pas parallélisées

Aller plus loin

- Traitement des configurations nécessitant de grandes quantités de RAM
  - Promotion de librairies classiques : ScaLAPACK, PETSc, etc...
  - Solutions de distribution des données « faites main »
- Compter sur la dissémination dans le laboratoire (mais il faut pousser quand même)
- Promouvoir d'autres services pour des problèmes inadaptés au parallélisme de cluster
  - Portages sur Grilles, etc...

Questions ?