Stage de fin de cycle

Master 2 recherche en informatique, des Concepts Aux Systèmes (COSY)

Option : Architectures à haute performance

Thème :

Étude de l'exclusion mutuelle pour les architectures à mémoire distribuée et Implantation de la directive CRITICAL dans l'outil STEP

Réalisé par :

KHALDI Dounia

Encadré par :

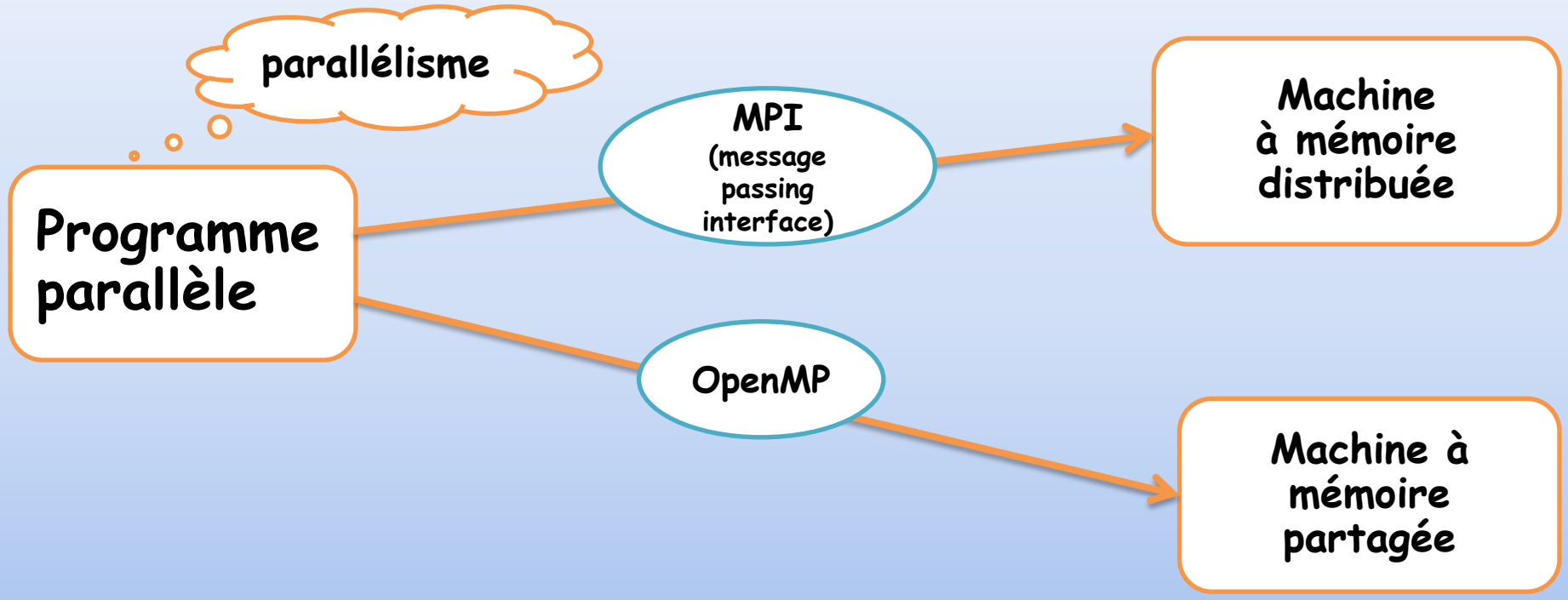
Frédérique Silber-Chaussumier

2009/2010

Plan

- *Introduction et objectif*
- *Travaux antérieurs*
- *STEP*
- *Section critique en OpenMP*
- *Implantation de la directive CRITICAL dans STEP*
- *Mesures de performance*
- *Conclusion et perspectives*

INTRODUCTION

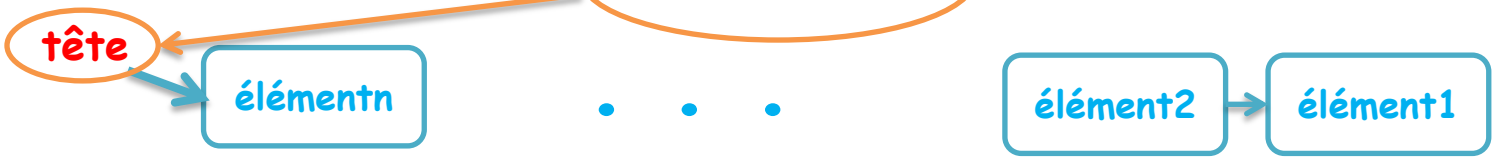


PROBLÉMATIQUE

MPI: contrôle fin des communications, mais code fragmenté difficile à lire
OpenMP: vision globale de l'application d'origine, mais restreint aux architectures à mémoire partagée

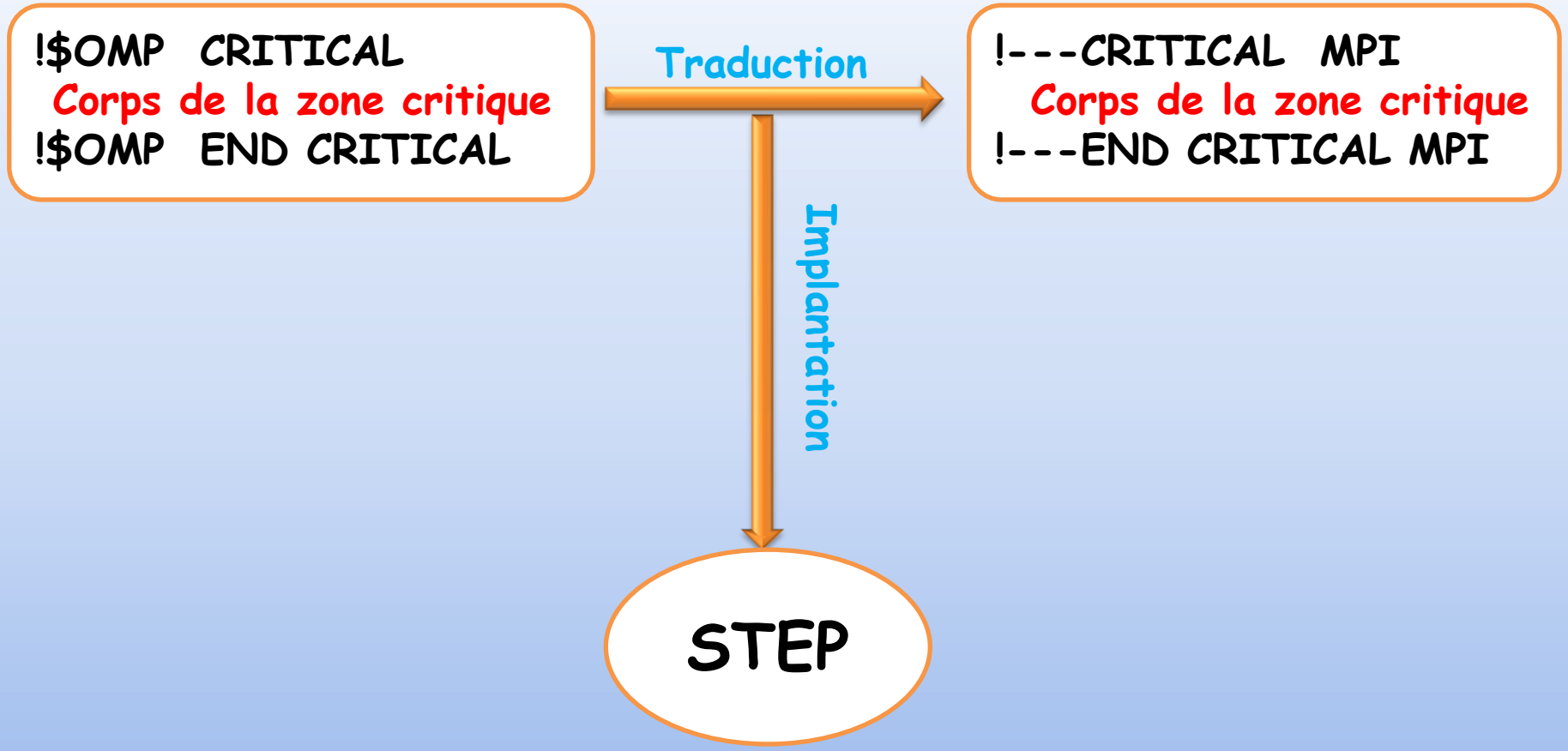
Transformations OpenMP → MPI

Programmes contenant des accès exclusifs à des données partagées



Traduction de la directive CRITICAL → CRITICAL
Modification de tête
END CRITICAL

OBJECTIF


!$OMP CRITICAL
Corps de la zone critique
!$OMP END CRITICAL
Traduction
!---CRITICAL MPI
Corps de la zone critique
!---END CRITICAL MPI
Implantation
STEP

TRAVAUX ANTÉRIEURS

CETUS: A source to source Compiler Infrastructure for C programs [BBD09]

- OpenMP vers MPI : exécution *in order* de la section critique
- OpenMP pour SDSM « software distributed shared memory » : CRITICAL→Reduction
- OpenMP pour GPGPU : CRITICAL → le host CPU

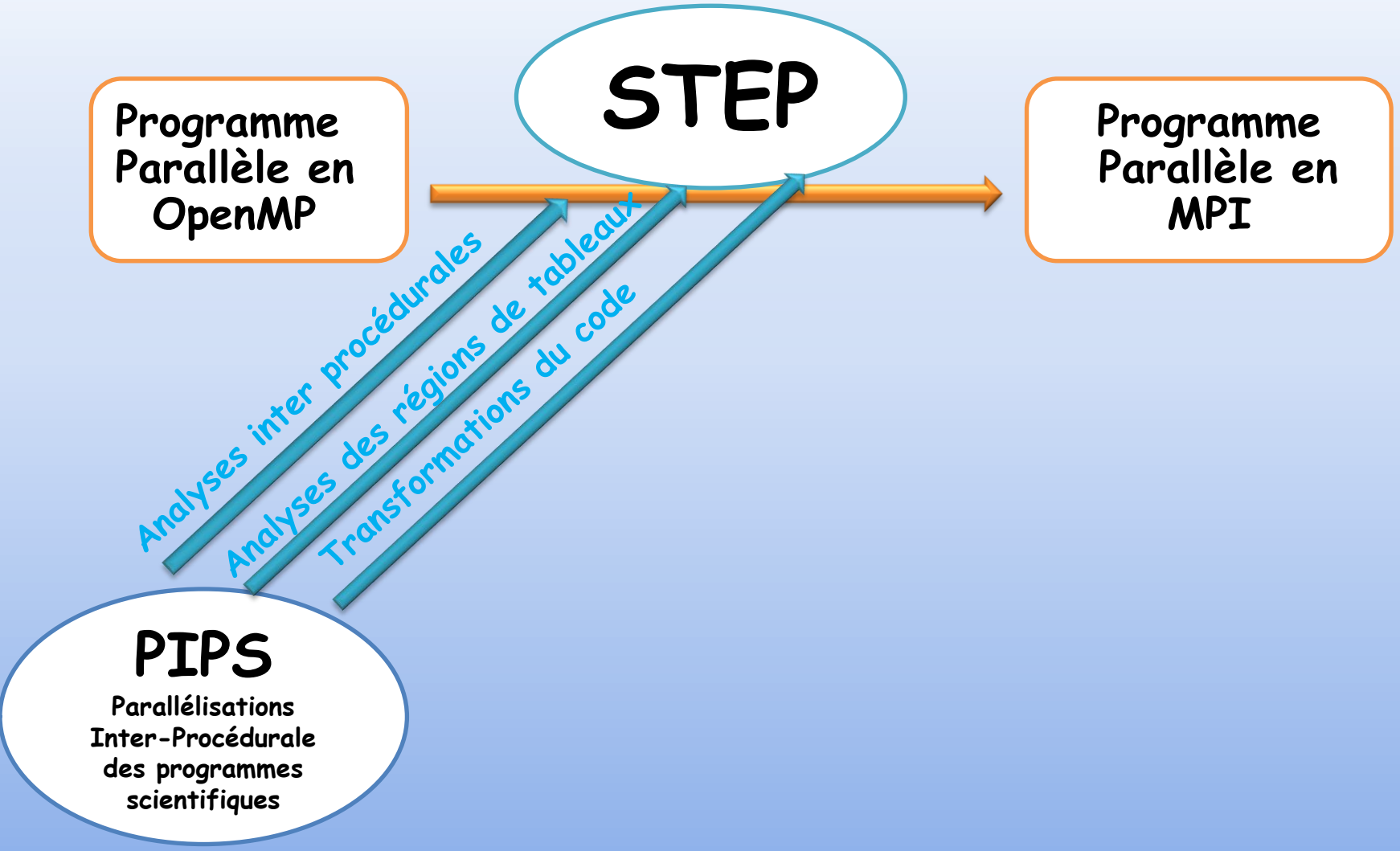
OpenMP vers GA (Global Arrays) [HCK03]

- Couche logiciel SDSM→ mémoire partagée
- primitives GA de verrous GA_LOCK

Intel Cluster OpenMP [Hoe06]

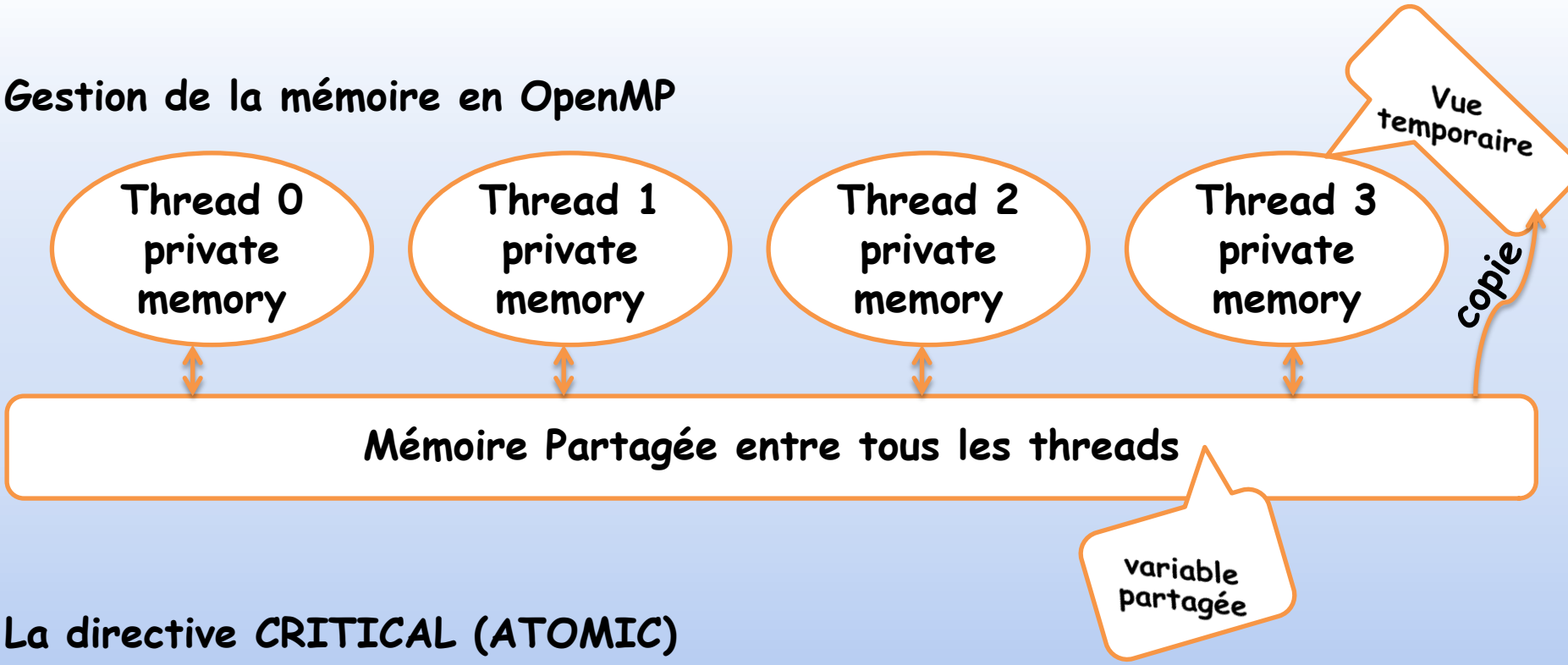
- Couche logiciel SDSM→ mémoire partagée
- Verrous → CRITICAL

STEP SYSTÈME DE TRANSFORMATION POUR L'EXÉCUTION PARALLÈLE



SECTION CRITIQUE EN OPENMP

Gestion de la mémoire en OpenMP



La directive CRITICAL (ATOMIC)

```
!$OMP CRITICAL [(nom)
        !--- Corps de la zone critique
!$OMP END CRITICAL [(nom)]
```

→

FLUSH

```
!$OMP CRITICAL [(nom)
        !--- Corps de la zone critique
!$OMP END CRITICAL [(nom)]
```

FLUSH

DIFFÉRENTS TYPES DE SECTIONS CRITIQUES

Réduction (scalaire ou tableau)

Réduction sur un scalaire

```
$!OMP PARALLEL DO REDUCTION (*:result)
DO i = 1, 10
        result = result * a(i)
END DO
$!OMP END PARALLEL DO
```

Réduction sur un tableau

```
$!OMP PARALLEL DO REDUCTION (*:result)
DO i = 1, 10
        DO j = 1, 10
                result[j] = result[j] * a
        END DO
END DO
$!OMP END PARALLEL DO
```

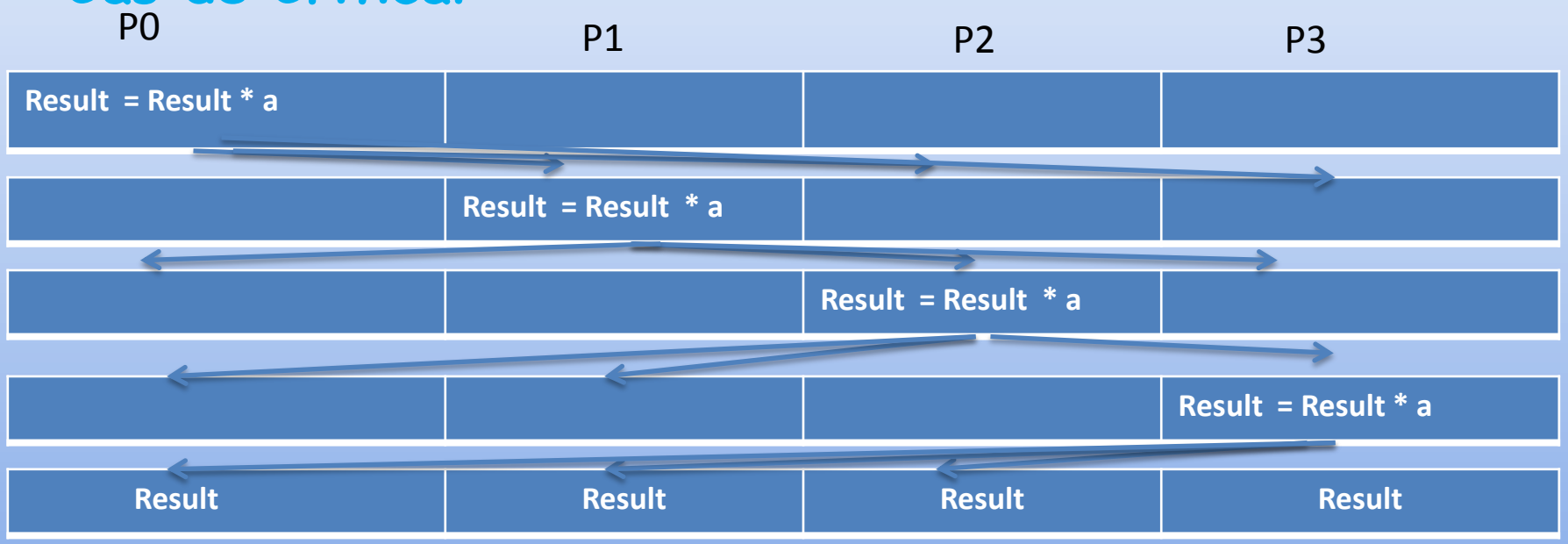
DIFFÉRENTS TYPES DE SECTIONS CRITIQUES

Réduction VS Critical : Représentation géométrique

```
$!OMP PARALLEL DO
DO i = 1, 4
        $!OMP CRITICAL
            result = result * a
        $!OMP END CRITICAL
END DO
$!OMP END PARALLEL DO
```

```
$!OMP PARALLEL DO REDUCTION (*:result)
        DO j = 1, 4
                result = result * a
        END DO
$!OMP END PARALLEL DO
```

Cas de Critical



DIFFÉRENTS TYPES DE SECTIONS CRITIQUES

Réduction VS Critical : Représentation géométrique

```
$!OMP PARALLEL DO
DO i = 1, 4
        $!OMP CRITICAL
            result = result * a
        $!OMP END CRITICAL
END DO
$!OMP END PARALLEL DO
```

```
$!OMP PARALLEL DO REDUCTION (*:result)
        DO j = 1, 4
                result = result * a
        END DO
$!OMP END PARALLEL DO
```

Cas de la réduction

P0	P1	P2	P3
Result 1 = Result1 * a	Result 2 = Result2 * a	Result 3 = Result3 * a	Result 4 = Result4 * a
Result 1= Result1 * Result2		Result3= Result3 * Result4	
Result = Result1 * Result3			
Result	Result	Result	Result

DIFFÉRENTS TYPES DE SECTIONS CRITIQUES

Mise à jour des structures de données partagées:

Opérations non-arithmétiques

```
#pragma omp critical
    {
        temp->next = head;
        head = temp;
    }
```

Mise à jour dans le cas régulier:
concerne tous les processus

Mise à jour dans le cas irrégulier:
concerne un sous ensemble de processus

```
If (monrang != 0)
    Exécuter la section critique
Endif
```

UTILISATION DES DIRECTIVES OPENMP CRITICAL ET ATOMIC DANS DES BENCHMARKS CLASSIQUES

Benchmark	Nombre de benchmarks	Nombre (et %) de critical (et atomic)	Répartition des types de sections critiques
NPB (NAS Parallel benchmark version 3.2)	11	7	100% en réduction de tableau
SPEC (Standard Performance Evaluation Corporation, version 3.2)	11	8	37,5% en réduction de scalaire, 25% en réduction de tableau, 37,5% en mise à jour (cas régulier)
OmpSrc (OpenMP source Version 2.0)	15	5	20% en réduction de scalaire, 40% en réduction de tableau, 40 % en mise à jour (cas régulier)
total	37	20	74,16% en réduction 25,84% en mise à jour

PROPOSITIONS D'IMPLANTATION DE LA DIRECTIVE CRITICAL

75% CRITICAL cas de la réduction

Mise à jour des structures partagées: une solution pour

le cas régulier
Et le cas irrégulier

IMPLANTATION DE LA DIRECTIVE CRITICAL
CAS DE LA RÉDUCTION

```
!$omp parallel default(shared)
!$omp& private(k,kk,t1,t2,t3,t4,i,ik,x,x1,x2,l,qq)
...
!$omp do reduction(+:sx,sy)
    do 150 k = 1, np
                    ...
      do 140 i = 1, nk
                    ...
          x1 = 2.d0 * x(2*i-1)  -  1.d0
          x2 = 2.d0 * x(2*i)  - 1.d0
          t3=(x1*t2)
          t4=(x2*t2)
          l    = max(abs(t3), abs(t4))
          qq(l) = qq(l) + 1.d0
          sx   = sx + t3
          sy   = sy + t4
        endif
140     continue
150  continue
    do 155 i = 0, nq - 1
!$omp critical
        q(i) = q(i) + qq(i)
!$omp end critical
155  continue
!$omp end parallel
```

```
!$omp parallel default(shared)
!$omp& private(k,kk,t1,t2,t3,t4,i,ik,x,x1,x2,l)
!$omp do reduction(+:sx,sy,q)
    do 150 k = 1, np
                ...
      do 140 i = 1, nk
                        ...
          l    = max(abs(t3), abs(t4))
          q(l) = q(l) + 1.d0
          sx   = sx + t3
          sy   = sy + t4
140     continue
150  continue
!$omp end parallel
```

Code MPI correspondant
Reduction→MPI_reduce()

IMPLANTATION DE LA DIRECTIVE CRITICAL
CAS DE MISE À JOUR DE STRUCTURES PARTAGÉES

Choix de la technique centralisée

- Algorithmes de l'exclusion mutuelle dans un système distribué [RAY86]
  Centralisé, distribué, à base de jeton
- Adaptation de la technique centralisée pour les architectures à mémoire distribuée
  - Cas régulier et irrégulier
    - simple

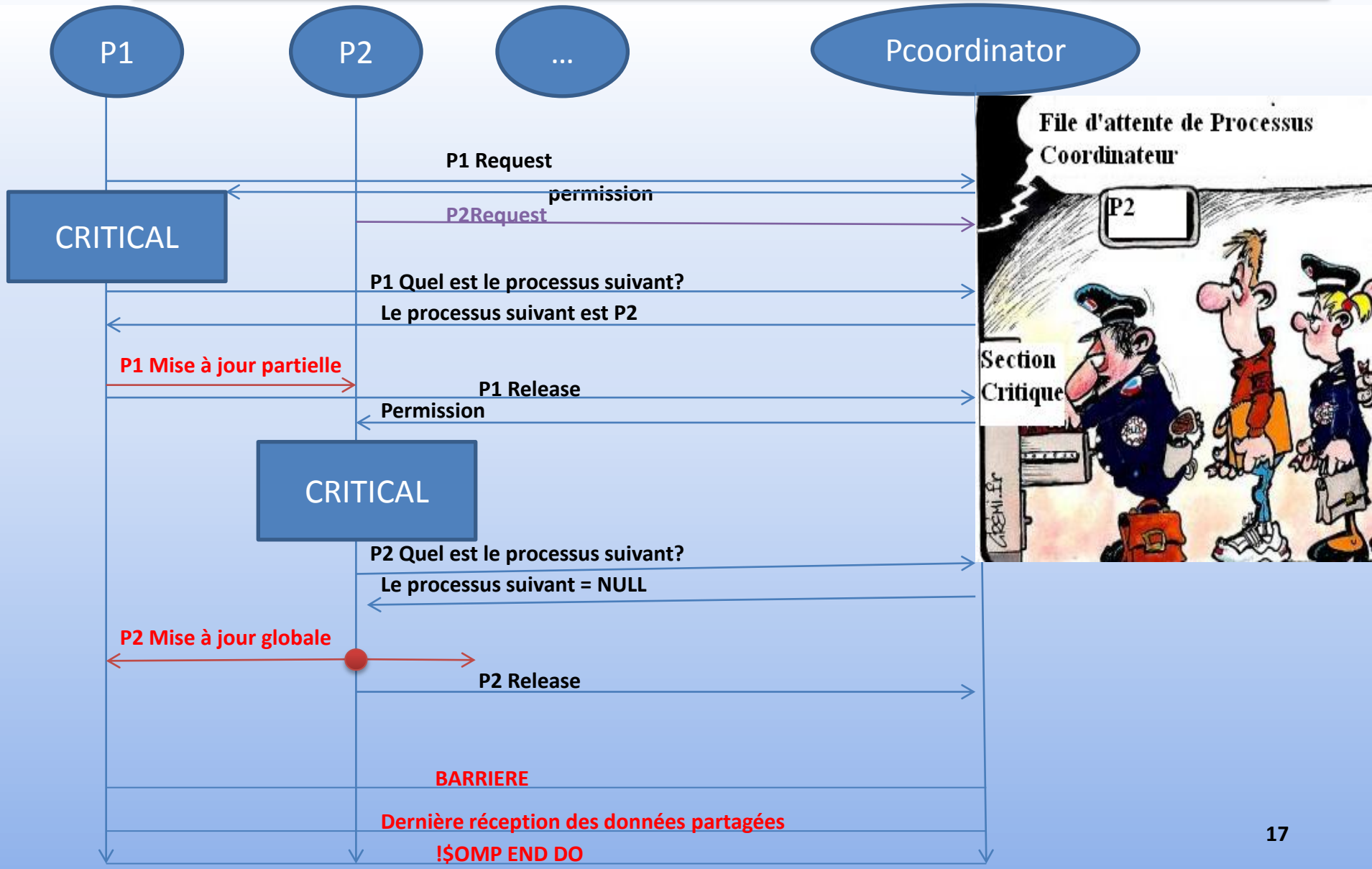
Choix d'implémentation avec un processus coordinateur Pcoord

- Au démarrage de l'application
- Une file d'attente par section critique
- Allocation des ressources: MPI_Comm_Spawn

Choix des communications et synchronisations

- Mise à jour partielle → processus suivant
- Mise à jour globale : Tous les processus du système

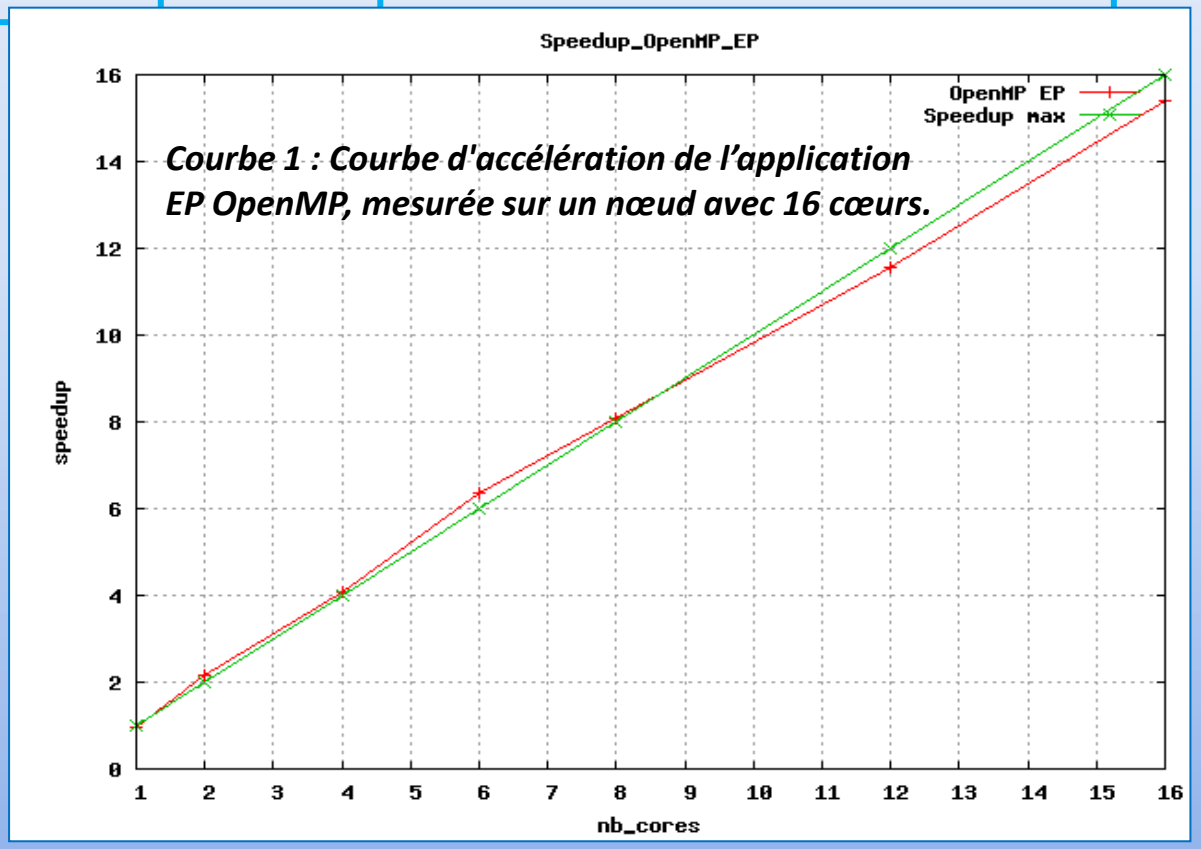
IMPLANTATION DE LA DIRECTIVE CRITICAL
CAS DE MISE À JOUR DE STRUCTURES PARTAGÉES



MESURES DE PERFORMANCES: (EP-NPB: NP=2**12,NQ=10)

Node	Node Type	Server	Nb CPUs	Characteristics
NV4	Compute	R480-E1	16	4 sockets quadri-core à 2,93 GHz 48 GB Mem 1 x 146 GB sas
to	…	…		
NV19	Compute	R480-E1	16	4 sockets quadri-core à 2,93 GHz 48 GB Mem 1 x 146 GB sas

```
!$omp parallel default(shared)
!$omp& private(k,kk,t1,t2,t3,t4,i,ik,x,x1,x2,l,qq)
...
!$omp do reduction(+:sx,sy)
    do 150 k = 1, np
                    ...
      do 140 i = 1, nk
                    ...
          l   = max(abs(t3), abs(t4))
          qq(l) = qq(l) + 1.d0
          sx   = sx + t3
          sy   = sy + t4
        endif
 140    continue
 150  continue
    do 155 i = 0, nq - 1
!$omp critical
        q(i) = q(i) + qq(i)
!$omp end critical
 155  continue
!$omp end parallel
```



Courbe 1 : Courbe d'accélération de l'application EP OpenMP, mesurée sur un nœud avec 16 cœurs.

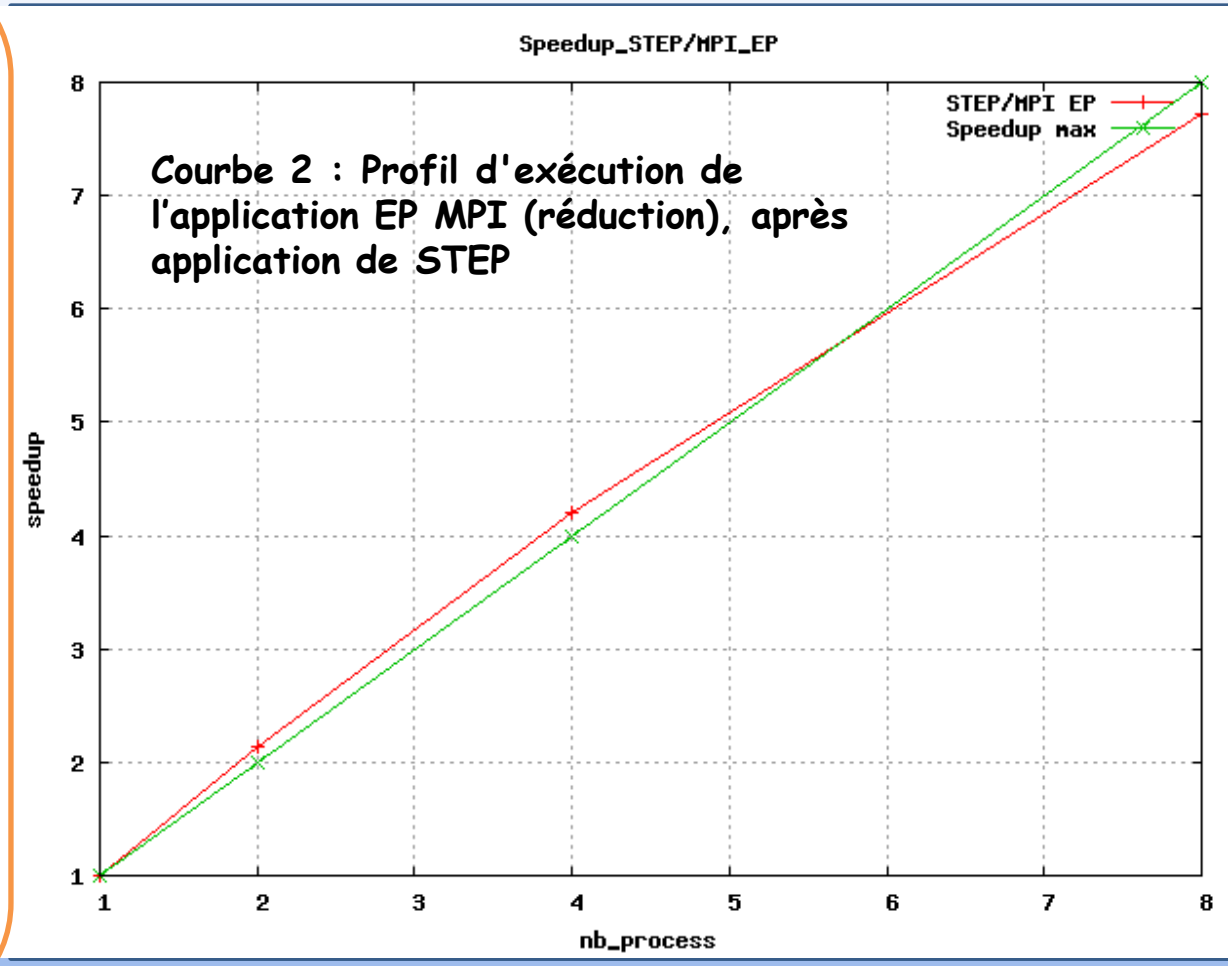
MESURES DE PERFORMANCES
CAS DE LA RÉDUCTION

```
!$omp parallel default(shared)
!$omp& private(k,kk,t1,t2,t3,t4,i,ik,x,x1,x2,l,qq)
...
!$omp do reduction(+:sx,sy)
    do 150 k = 1, np
                ...
      do 140 i = 1, nk
                ...
          x1 = 2.d0 * x(2*i-1)  -  1.d0
          x2 = 2.d0 * x(2*i)  - 1.d0
          t3=(x1*t2)
          t4=(x2*t2)
          l    = max(abs(t3), abs(t4))
          qq(l) = qq(l) + 1.d0
          sx   = sx + t3
          sy   = sy + t4
        endif
 140     continue
 150  continue
    do 155 i = 0, nq - 1
!$omp critical
        q(i) = q(i) + qq(i)
!$omp end critical
 155  continue
!$omp end parallel
```

```
!$omp parallel default(shared)
!$omp& private(k,kk,t1,t2,t3,t4,i,ik,x,x1,x2,l)
!$omp do reduction(+:sx,sy,q)
    do 150 k = 1, np
                ...
      do 140 i = 1, nk
                ...
          l    = max(abs(t3), abs(t4))
          q(l) = q(l) + 1.d0
          sx   = sx + t3
          sy   = sy + t4
 140     continue
 150  continue
!$omp end parallel
```

MESURES DE PERFORMANCES
CAS DE LA RÉDUCTION

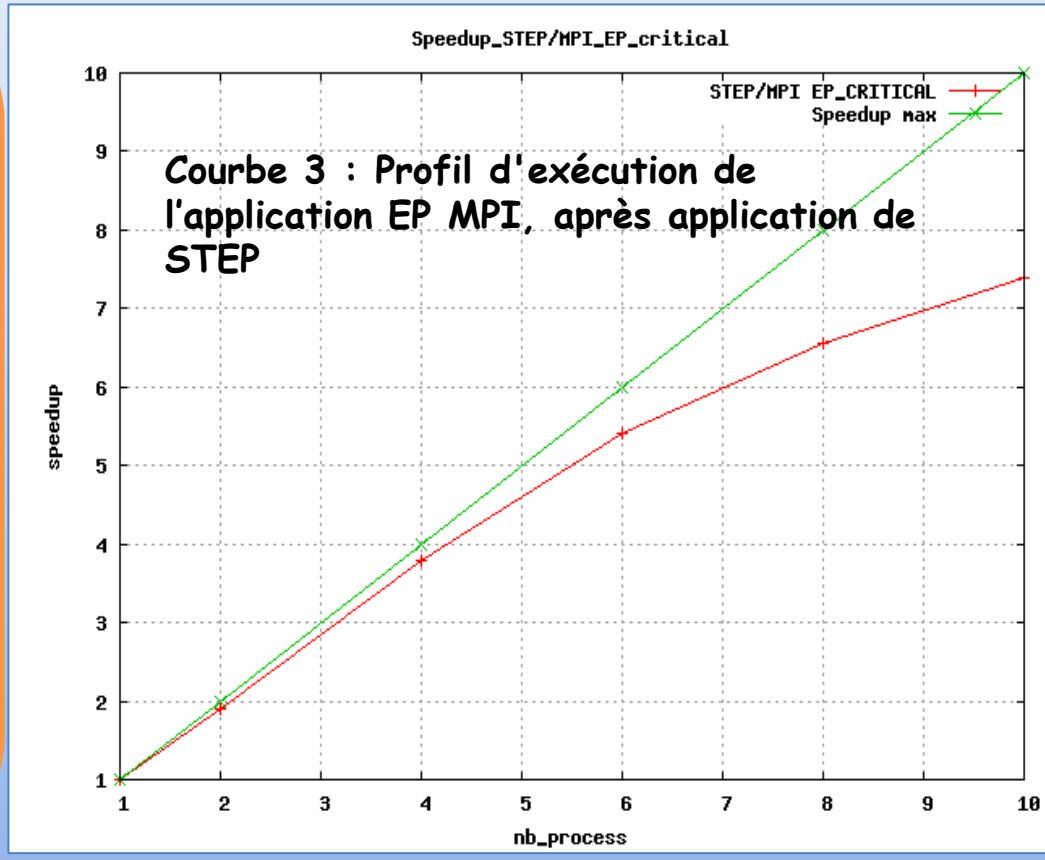
```
!!
!! file for EMBAR_PAR1_DO150_MPI.f
!!
    SUBROUTINE EMBAR_PAR1_DO150_MPI(K_DUMMY,
   &K_L, K_U, KK, K_OFFSET
   &, T1, S, T2, AN, I, T3, IK, NK, A, X, X1, X2, T4, L, NQ,
   & Q, SX, SY)

CALL STEP_CONSTRUCT_BEGIN(STEP_DO)
CALL STEP_INITREDUCTION(Q, STEP_SUM, STEP_REAL8)
 CALL STEP_GET_COMMSIZE(STEP_COMM_SIZE)
 CALL STEP_COMPUTE_LOOPSLICES(K_L, K_U, 1,
   & STEP_COMM_SIZE)
 DO 150 K = STEP_K_LOW, STEP_K_UP
      ...
DO 140 I = 1, NK
 ...
      L = MAX(ABS(T3), ABS(T4))
          Q(L) = Q(L)+1.D0
            SX = SX+T3
          SY = SY+T4
140     CONTINUE
150     CONTINUE
CALL STEP_REDUCTION(Q)
CALL STEP_CONSTRUCT_END(STEP_DO)
END
```



MESURES DE PERFORMANCES CAS DE CRITICAL

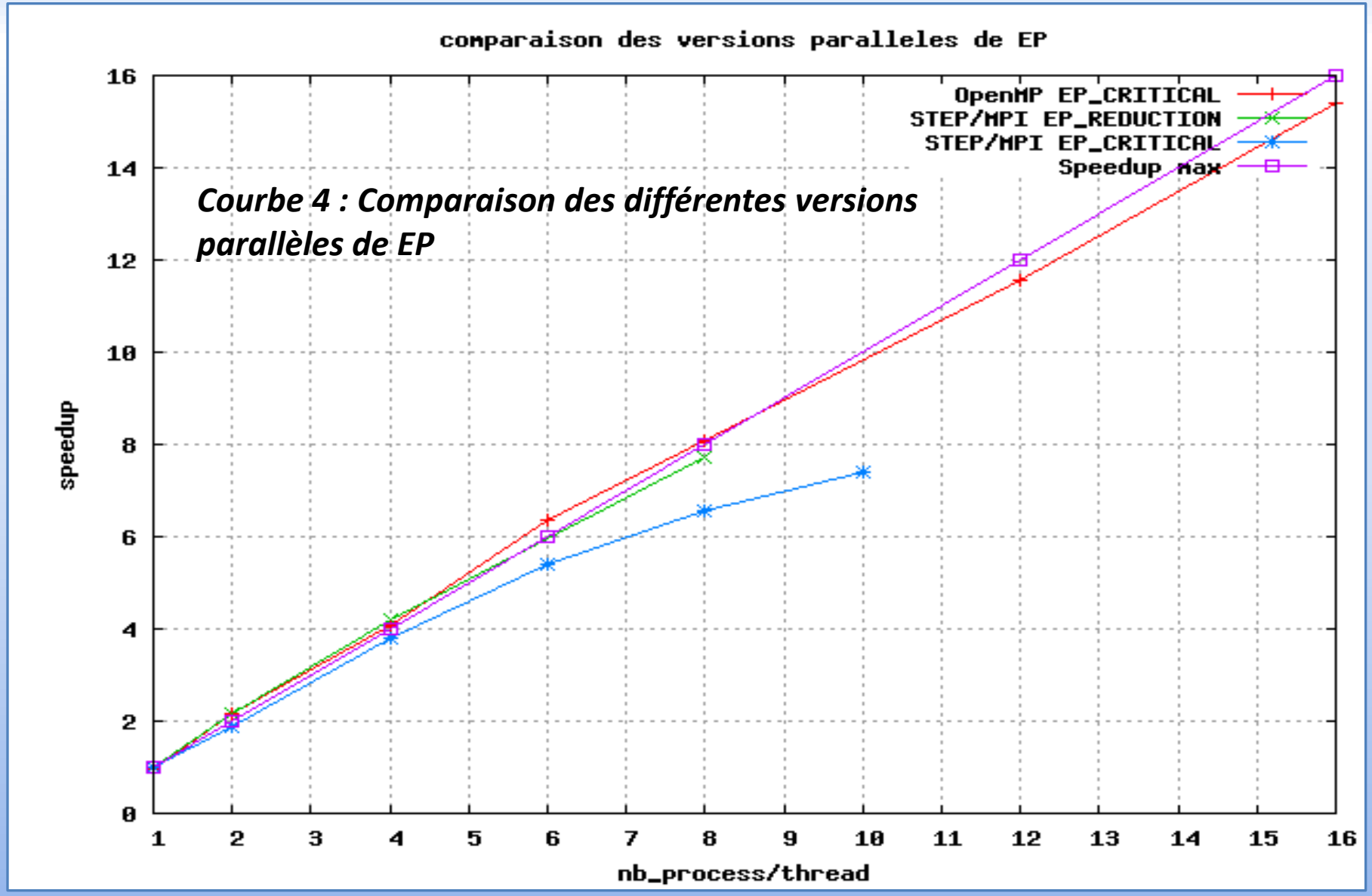
Nom	Jaguar (IT Sud Paris)
Nœuds	15
CPU	4 processeurs par nœud
Mémoire	16 GB de RAM par processeur

```
CALL STEP_CONSTRUCT_BEGIN(STEP_CRITICAL)
C     Compute send regions...
      CALL STEP_CRITICAL_REQUEST(0)
      CALL STEP_CRITICAL_GET_CURRENTUPTODATESCALAR
      (SUM)
      CALL STEP_WAITALL

C     Where work is done...
      Critical body

CALL STEP_CRITICAL_GET_NEXTPROCESS(STEP_CRITICAL)
CALL STEP_CRITICAL_SET_CURRENTUPTODATESCALAR(SUM)
CALL STEP_CRITICAL_RELEASE(STEP_CRITICAL)
CALL STEP_CONSTRUCT_END(STEP_CRITICAL)
END DO
CALL STEP_BARRIER
CALL STEP_CRITICAL_GET_FINALUPTODATESCALAR(SUM)
CALL STEP_WAITALL
```



Courbe 3 : Profil d'exécution de l'application EP MPI, après application de STEP

MESURES DE PERFORMANCES: (EP-NPB: NP=2**12,NQ=10)



Courbe 4 : Comparaison des différentes versions parallèles de EP

CONCLUSION

- **Adaptation de la section critique pour les architectures à mémoire distribuée**

- **Directive CRITICAL (OpenMP → MPI)**
  - Benchmarks: NPB, SPEC, OmpSrc
  - Algorithmes distribués (technique centralisée)

- **Implémentation dans l'outil STEP (compilateur PIPS)**
  - Cas de la réduction → améliorer les performances
  - Cas de critical : Cas *régulier* et *non régulier*

- **Comparaison avec les travaux antérieurs**
  - Cetus: OpenMP vers MPI → exécution *in order* de la section critique

PERSPECTIVES

Perspectives générales

- **OpenMP ou MPI ?**
  - parallélisme des programmes à grain fin ou à gros grain
  - Grain fin → Surcoût important des communications
- **Prise en charge des langages F95, C**

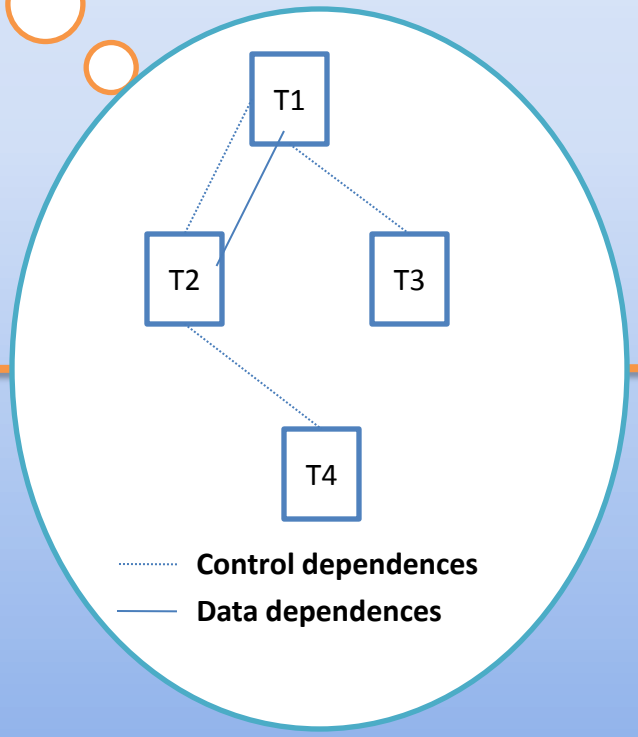
Perspectives spécifiques CRITICAL

- **Détection automatique des réductions de tableau**

AUTOMATIC TASK PARTITIONING OF PARALLEL PROGRAMS FOR MULTIPROCESSORS SYSTEM

- Execution time
- Parallelism
- Communication Cost
- Load balancing
- Code size
- Data Type
- Data dependences
- Memory constraints

Embedded real-time multimedia application



Heterogeneous multiprocessor (MPSoC)

MERCI

RÉFÉRENCES

- **[BBD09]** H.Bae, L.Bachega, C.Dave. "Cetus: A source to source compiler infrastructure for multicores ». Purdue university. 2009.

- **[HCK03]** L. Huang, B. Chapman, and R. Kendall. "OpenMP for Clusters". In Fifth European Workshop on OpenMP, EWOMP. 2003.

- **[Hoe06]** J. Hoeflinger. "Extending OpenMP to Clusters". Technical report, Intel Corporation. 2006.

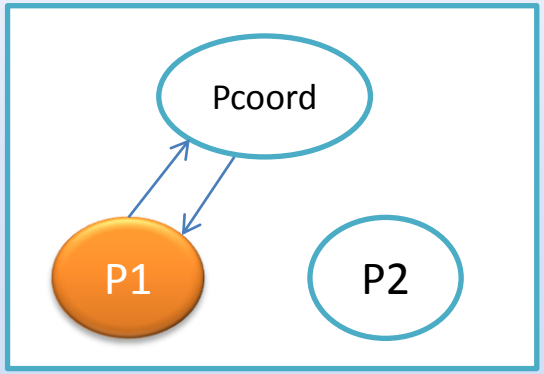
- **[RAY86]** M. RAYNAL. "Algorithms for mutual exlusion". The MIT Press. 1986

ANNEXES

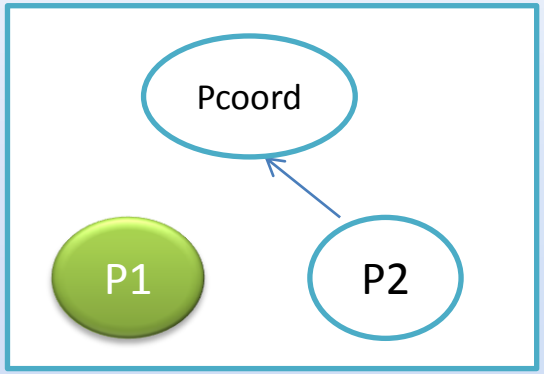
ALGORITHMES DE L'EXCLUSION MUTUELLE DANS UN SYSTÈME DISTRIBUÉ [RAYNAL]

Algorithme centralisé

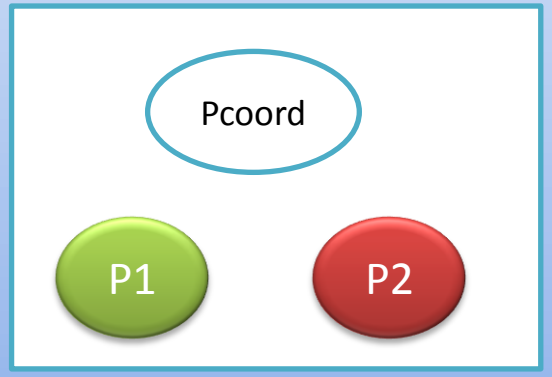
Configuration0



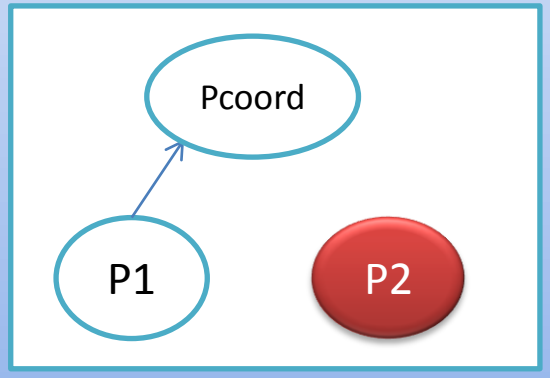
C1:P1 exécute la section critique



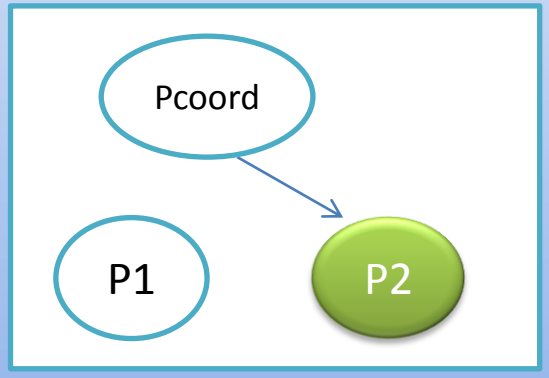
C2: P2 est en attente de P0



C3: P1 libère la section critique



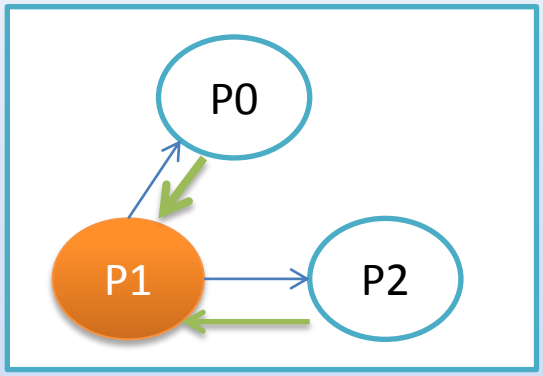
C4: P0 débloque P2



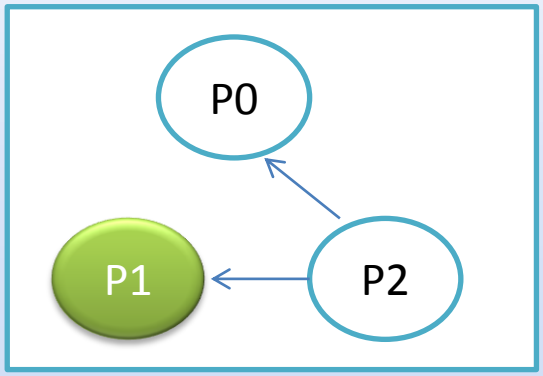
ALGORITHMES DE L'EXCLUSION MUTUELLE DANS UN SYSTÈME DISTRIBUÉ [RAYNAL]

Algorithme distribué

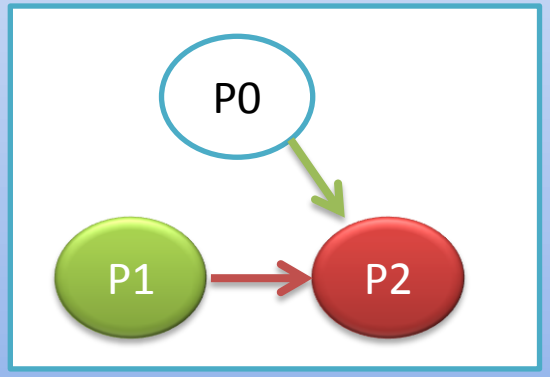
Configuration0



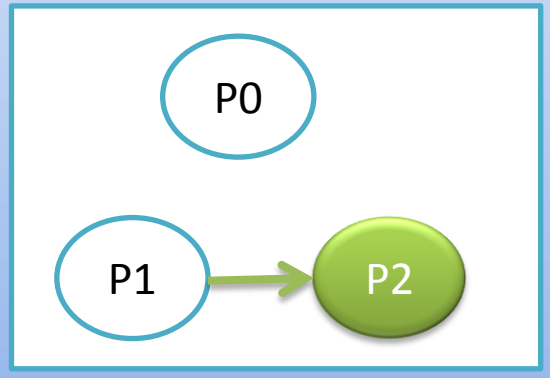
C1:P1 exécute la section critique



C2: P2 est en attente de P1



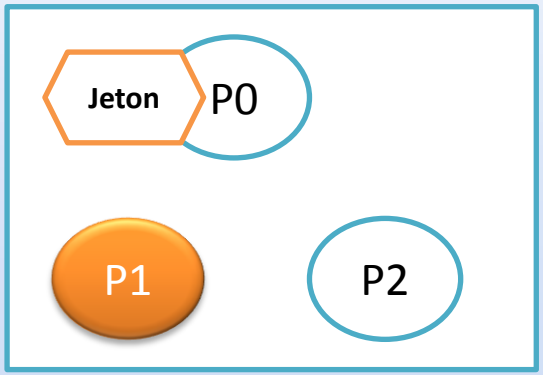
C3: P1 débloque P2



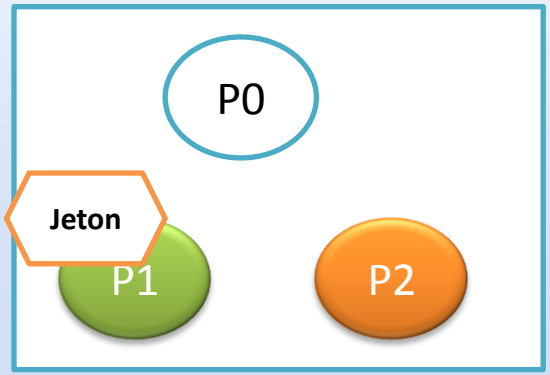
ALGORITHMES DE L'EXCLUSION MUTUELLE DANS UN SYSTÈME DISTRIBUÉ [RAYNAL]

Algorithme à base de jeton (Token Ring)

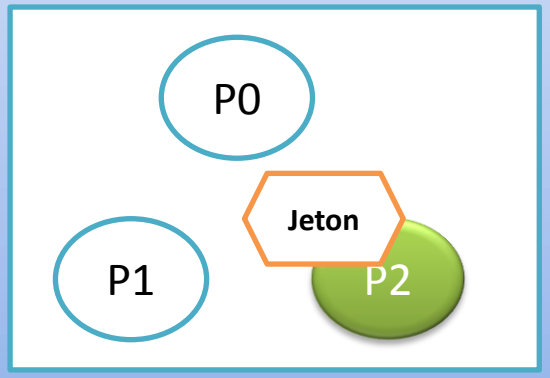
Configuration0



C1:P1 exécute la section critique



C2: P2 obtient le jeton



C3: P2 libère le jeton