PT-SCOTCH: Un outil pour la renumérotation parallèle efficace de grands graphes dans un contexte multi-niveaux

Cédric Chevalier et François Pellegrini
Projet SCALAPPLIX

LaBRI – INRIA Futurs
Université Bordeaux I

Groupe de Travail – 24 Octobre 2006

Sommaire

Partitionnement de graphes

Procédé qui consiste à répartir les sommets d'un graphe en un nombre donné d'ensembles, en respectant deux contraintes :

1. Contrainte de frontière : la taille de l'interface entre les parties doit être la plus petite possible
2. Contrainte d'équilibre : tous les ensembles doivent être de même poids

Utilisations du partitionnement de graphes

- Nombreuses applications : équilibrage de charge, renumérotation de matrices, conception de circuits VLSI, bio-informatique, ...
- Plusieurs partitionneurs séquentiels existent déjà
  $\Longrightarrow$ SCOTCH, METIS
- Mais la taille des problèmes ne cesse de grandir
  $\Longrightarrow$ Besoin d'un partitionneur parallèle de graphes car les plus gros graphes ne logent pas dans la mémoire d'un ordinateur séquentiel
  - Le projet PT-SCOTCH (*Parallel Threaded* SCOTCH)
  - Actuellement concentré sur la renumérotation de matrices (problème des bipartitionnement récursif de graphes)

INRIA FUTURS

Utilisations du partitionnement de graphes

- Nombreuses applications : équilibrage de charge, renumérotation de matrices, conception de circuits VLSI, bio-informatique, ...
- Plusieurs partitionneurs séquentiels existent déjà
  $\Longrightarrow$ SCOTCH, METIS
- Mais la taille des problèmes ne cesse de grandir
  $\Longrightarrow$ Besoin d'un partitionneur parallèle de graphes car les plus gros graphes ne logent pas dans la mémoire d'un ordinateur séquentiel
  - Le projet PT-SCOTCH (*Parallel Threaded* SCOTCH)
  - Actuellement concentré sur la renumérotation de matrices (problème des bipartitionnement récursif de graphes)

INRIA FUTURS

Utilisations du partitionnement de graphes

- Nombreuses applications : équilibrage de charge, renumérotation de matrices, conception de circuits VLSI, bio-informatique, ...
- Plusieurs partitionneurs séquentiels existent déjà
  $\Longrightarrow$ SCOTCH, METIS
- Mais la taille des problèmes ne cesse de grandir
  $\Longrightarrow$ Besoin d'un partitionneur parallèle de graphes car les plus gros graphes ne logent pas dans la mémoire d'un ordinateur séquentiel
  - Le projet PT-SCOTCH (*Parallel Threaded* SCOTCH)
  - Actuellement concentré sur la renumérotation de matrices (problème des bipartitionnement récursif de graphes)

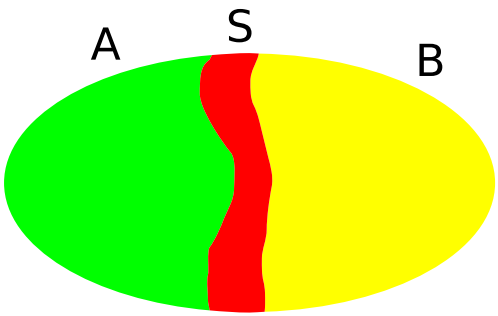
INRIA FUTURS

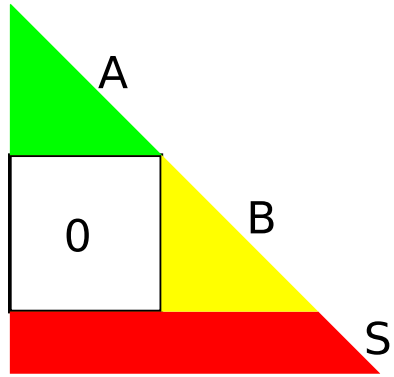
Renumérotation de matrices

- Lors de la résolution de systèmes linéaires creux avec des méthodes directes, des termes non nuls sont créés durant la factorisation ($A \longrightarrow LL^t$, $A \longrightarrow LDL^t$ ou $A \longrightarrow LU$)
- Le remplissage dépend de l'ordre des inconnues
  $\Longrightarrow$ Besoin de fournir des renumérotations qui limitent le remplissage
- La renumérotation de graphes dans SCOTCH est faite en utilisant les Dissections Emboîtées dans un contexte Multi-niveaux
- Métrique de la qualité de la renumérotation : OPC, le nombre d'opérations lors de la factorisation de Cholesky (nombre total d'additions, multiplications et divisions)

Renumérotation de matrices avec les Dissections Emboîtées

- Principe (George 1973)
  - Trouver un séparateur-sommet du graphe
  - Numéroter les sommets du séparateur avec les plus grands numéros disponibles
  - Appliquer récursivement aux deux sous-graphes séparés





- Intérêts
  - Induit des décompositions par blocs de grande qualité
  - Augmente l'indépendance entre les calculs
    ↪ très utile pour une factorisation parallèle (PaStiX)

L'environnement Multi-niveaux

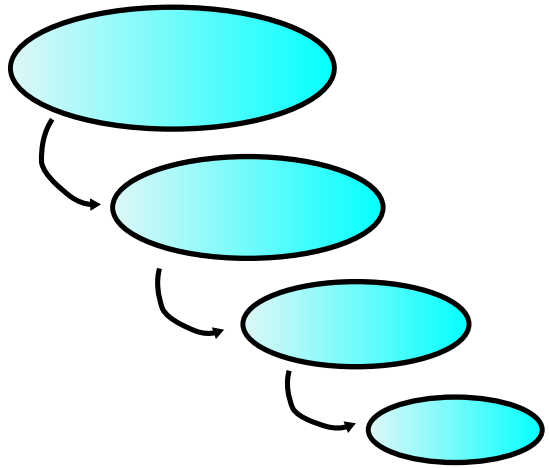
Une approche classique pour améliorer la qualité des partitions (Barnard et Simon, 1994)

Trois étapes

1. Contraction (Coarsening phase)
2. Séparation initiale
3. Expansion (Uncoarsening phase)

Diminue le nombre de sommets en regroupant des paires de sommets voisins

$\Longrightarrow$ À chaque pas, obtention d'un graphe plus petit avec la même topologie



L'environnement Multi-niveaux

Une approche classique pour améliorer la qualité des partitions (Barnard et Simon, 1994)

Trois étapes

1. Contraction (Coarsening phase)
2. Séparation initiale
3. Expansion (Uncoarsening phase)

Applique une heuristique locale pour calculer une partition du plus petit graphe

Typiquement, la taille du graphe le plus grossier est d'environ 100 sommets. De bonnes partitions initiales peuvent donc être calculées à bas coût.

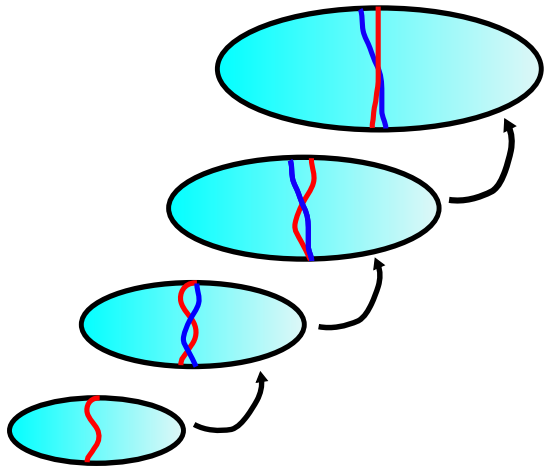
L'environnement Multi-niveaux

Une approche classique pour améliorer la qualité des partitions (Barnard et Simon, 1994)

Trois étapes

1. Contraction (Coarsening phase)
2. Séparation initiale
3. Expansion (Uncoarsening phase)

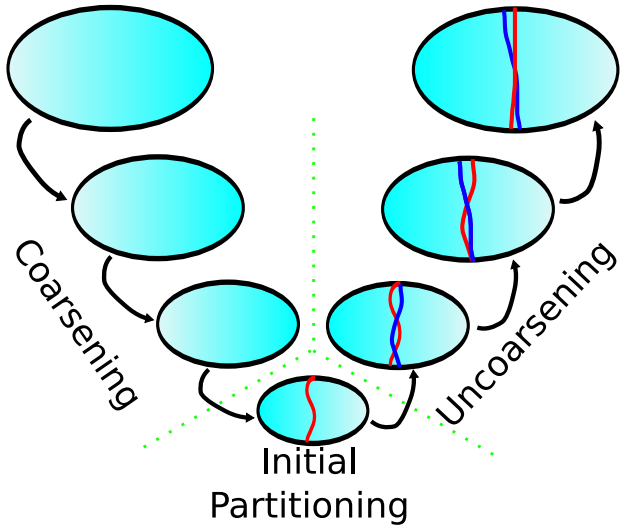
Projette la partition calculée du graphe contracté vers les graphes plus fins
Optimise localement en utilisant des heuristiques telles que Kernighan-Lin ou Fiduccia-Mattheyses (F.M.)



L'environnement Multi-niveaux

Trois étapes

1. Contraction (Coarsening phase)
2. Séparation initiale
3. Expansion (Uncoarsening phase)



Limites de ce modèle en parallèle

Les algorithmes multi-niveaux sont difficiles à paralléliser

- Problèmes durant la phase de contraction :
  - Les formulations parallèles des algorithmes de contraction séquentiels classiques nécessitent beaucoup de communications distantes pour apparier des sommets situés sur des processeurs différents
  - La qualité de contraction diminue quand les appariements locaux sont privilégiés
- L'étape d'expansion est même plus difficile :
  - Les meilleurs algorithmes de raffinement (comme F.M.) sont séquentiels par nature et ne se parallélisent pas bien

Limites de ce modèle en parallèle (2)

- Renumeroteurs parallèles disponibles (PARMETIS) avec un schéma multi-niveaux
- Désactivation des capacités de sortie des minima locaux ("hill-climbing") dans PARMETIS pour réduire les communications
  ↪ Méthode de gradient
  ↪ Très mauvais résultats quand le nombre de processeurs augmente

Notre objectif est de permettre le "hill-climbing" même en parallèle afin d'obtenir la même qualité que les meilleures méthodes séquentielles

INRIA FUTURS

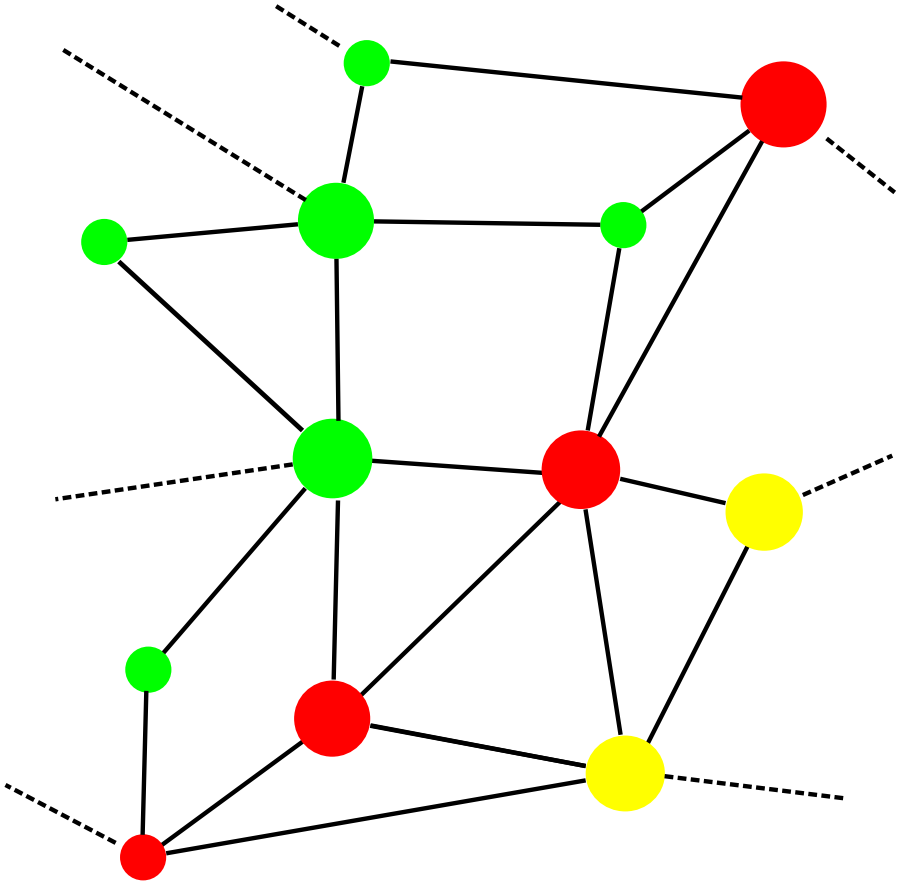
Sommaire

INRIA FUTURS

Observations lors du raffinement

Seuls les sommets situés dans une bande étroite autour du séparateur sont potentiellement déplacés durant l'expansion

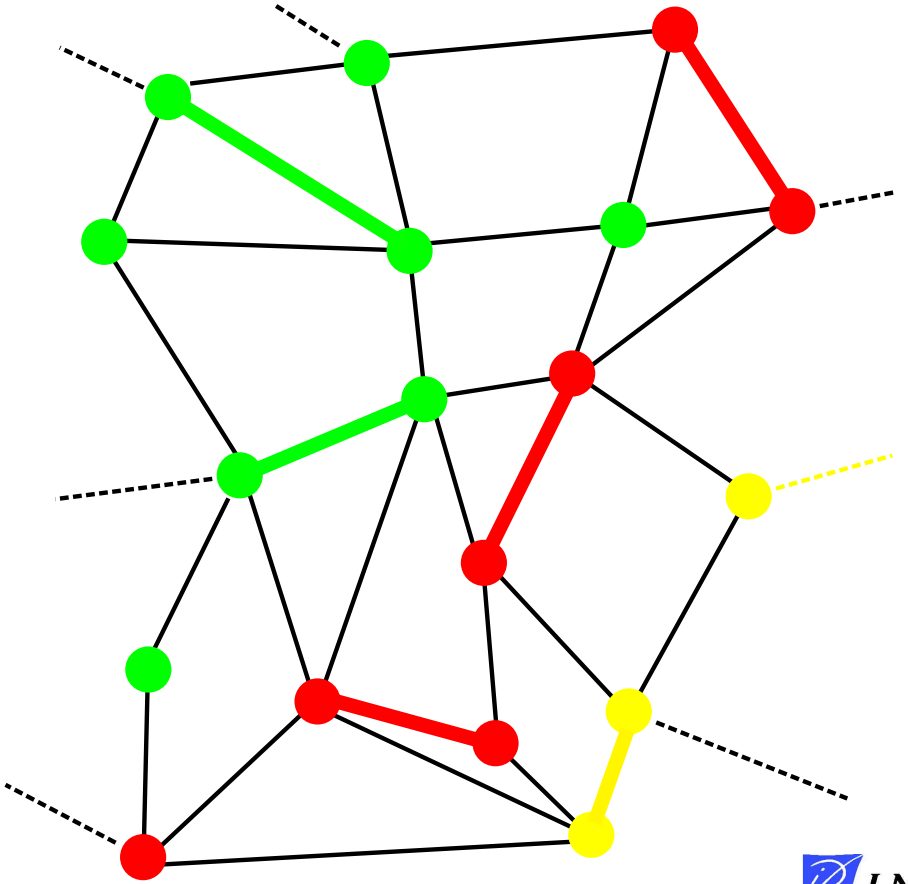
Le séparateur grossier :



Observations lors du raffinement

Seuls les sommets situés dans une bande étroite autour du séparateur sont potentiellement déplacés durant l'expansion

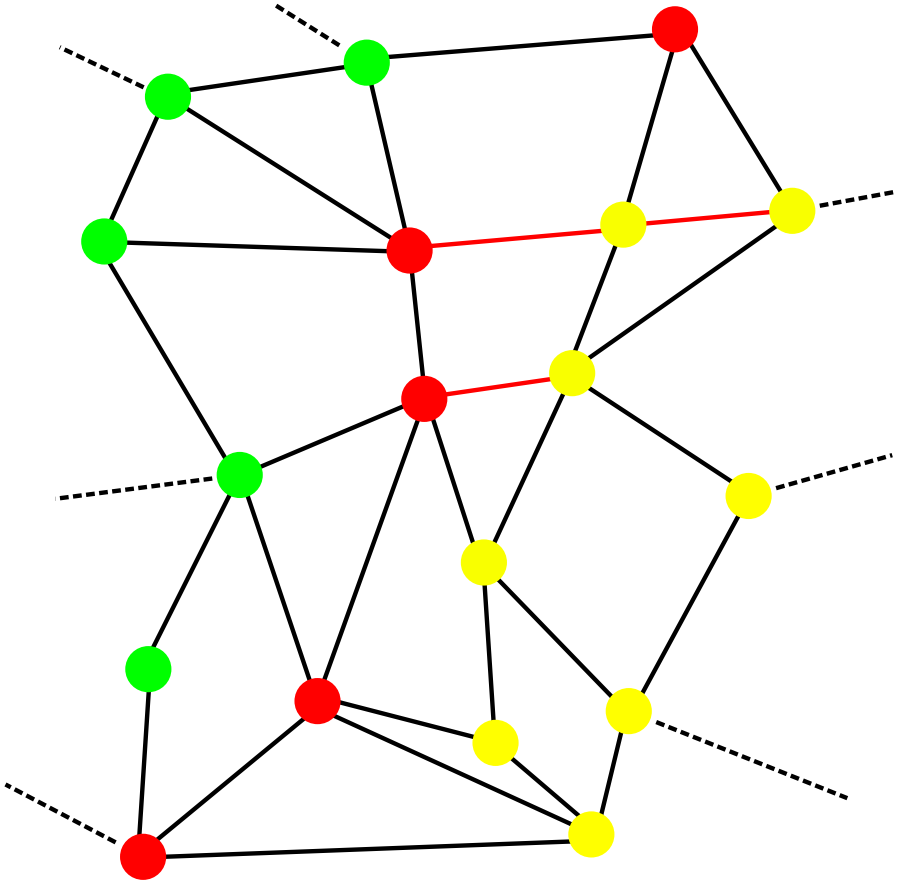
La projection du séparateur grossier :



Observations lors du raffinement

Seuls les sommets situés dans une bande étroite autour du séparateur sont potentiellement déplacés durant l'expansion

Le nouveau séparateur :



Introduction de la Bande

À partir d'observations expérimentales en regardant la distribution des sommets du séparateur sur le graphe fin après un raffinement local de type F.M.

Distance par rapport au séparateur grossier				
0	1	2	3	≥ 4
80%	17%	1.5%	$< 0.5\%$	$< 0.1\%$

Explication

Du fait que le graphe grossier est topologiquement proche du graphe plus fin, la projection de la partition calculée au niveau grossier est proche de celle du niveau plus fin et seuls quelques améliorations locales sont nécessaires.

Introduction de la Bande

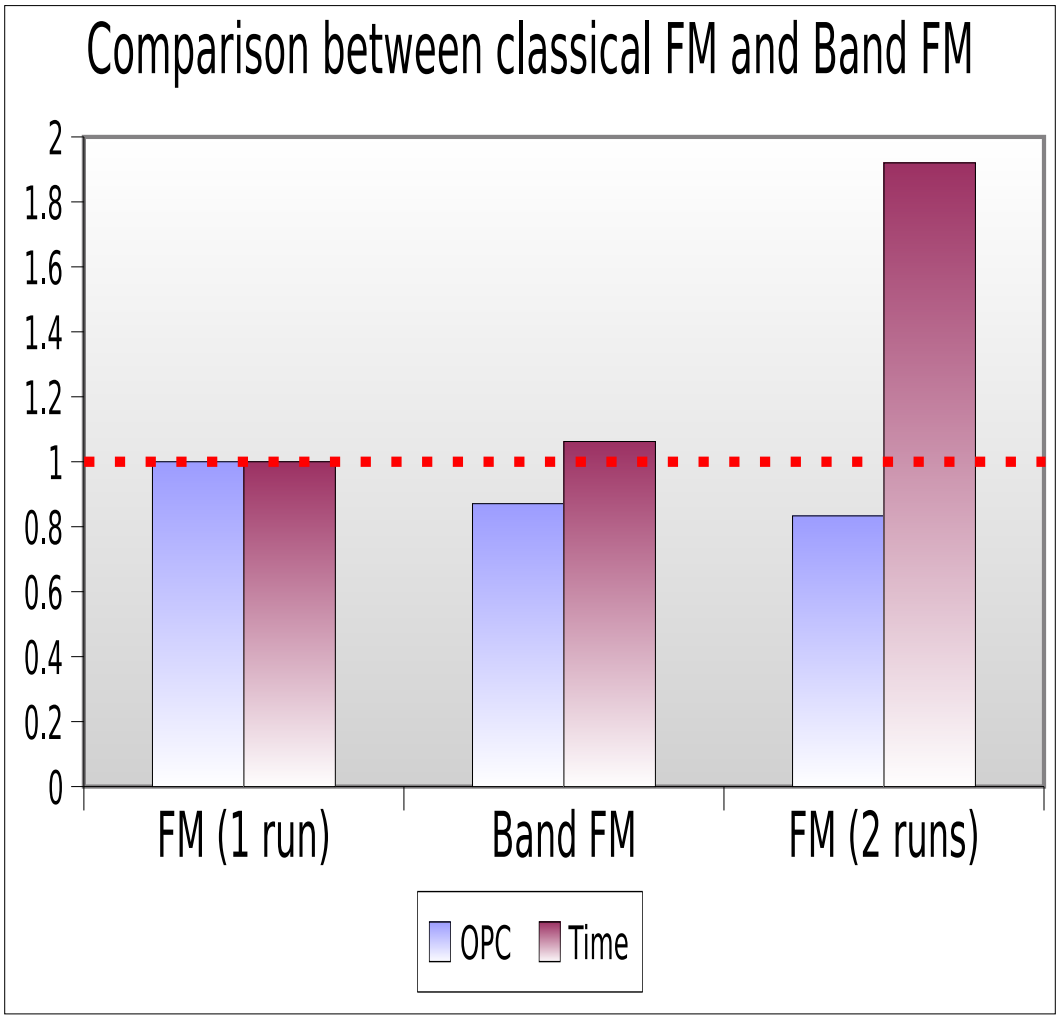
À partir d'observations expérimentales en regardant la distribution des sommets du séparateur sur le graphe fin après un raffinement local de type F.M.

Distance par rapport au séparateur grossier				
0	1	2	3	≥ 4
80%	17%	1.5%	$< 0.5\%$	$< 0.1\%$

Idée

Considérer seulement, pour le raffinement, les sommets qui sont susceptibles de bouger
$\Longrightarrow$ Extraire a priori un sous-graphe bande autour du séparateur grossier et n'appliquer les heuristiques d'optimisation qu'à lui

Résultats expérimentaux avec la bande FM



La bande FM :

1. n'augmente pas le temps d'éxécution de manière significative
2. ne diminue pas la qualité de la partition mais au contraire l'améliore

Conséquences de la bande

Conséquences

Il est possible de réduire la taille du problème pour l'optimisation locale durant l'expansion, d'où :

- Réduction de la quantité de mémoire nécessaire
- Réduction du temps nécessaire, comme dans SCOTCH on utilisait par défaut 2 passes

$Size_{band} = O(Size_{separator}) = O(n^{\frac{2}{3}})$ pour les graphes 3D, où n est le nombre de sommets du graphe

Pour des graphes 3D avec 10^9 sommets, la taille de la bande est $\approx 10^6$ donc le raffinement peut même être calculé en séquentiel

Sommaire

Parallélisation du bipartitionnement multi-niveaux de graphes

Notre algorithme de bipartitionnement parallèle exploite trois niveaux de parallélisme

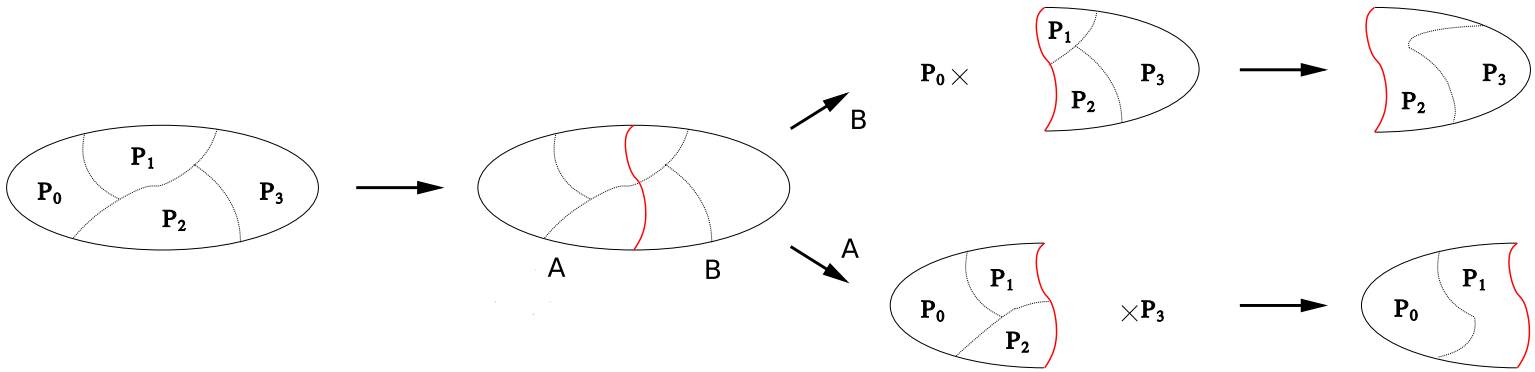
1. Dans le processus de Dissections Emboîtées lui-même
2. Dans la phase de contraction de l'algorithme multi-niveaux
3. Dans le processus de raffinement lors de l'expansion

Parallélisation de la Dissection Emboîtée

- Parallélisme à gros grain
- Tous les sous-graphes d'un même niveau de dissection sont calculés en même temps sur des ensembles séparés de processeurs
- Après que le séparateur a été calculé, les deux sous-graphes séparés sont repliés et redistribués sur deux sous-ensembles des processeurs disponibles
  Possibilité de replier sur un nombre quelconque de processeurs (pas seulement une puissance de deux)
  $\Longrightarrow$ Meilleure localité des données
  $\Longrightarrow$ Les deux sous-arbres sont séparés non seulement de manière logique mais aussi de manière physique, ce qui aide à réduire la congestion réseau

Parallélisation de la Dissection Emboîtée (2)

Schéma du processus :



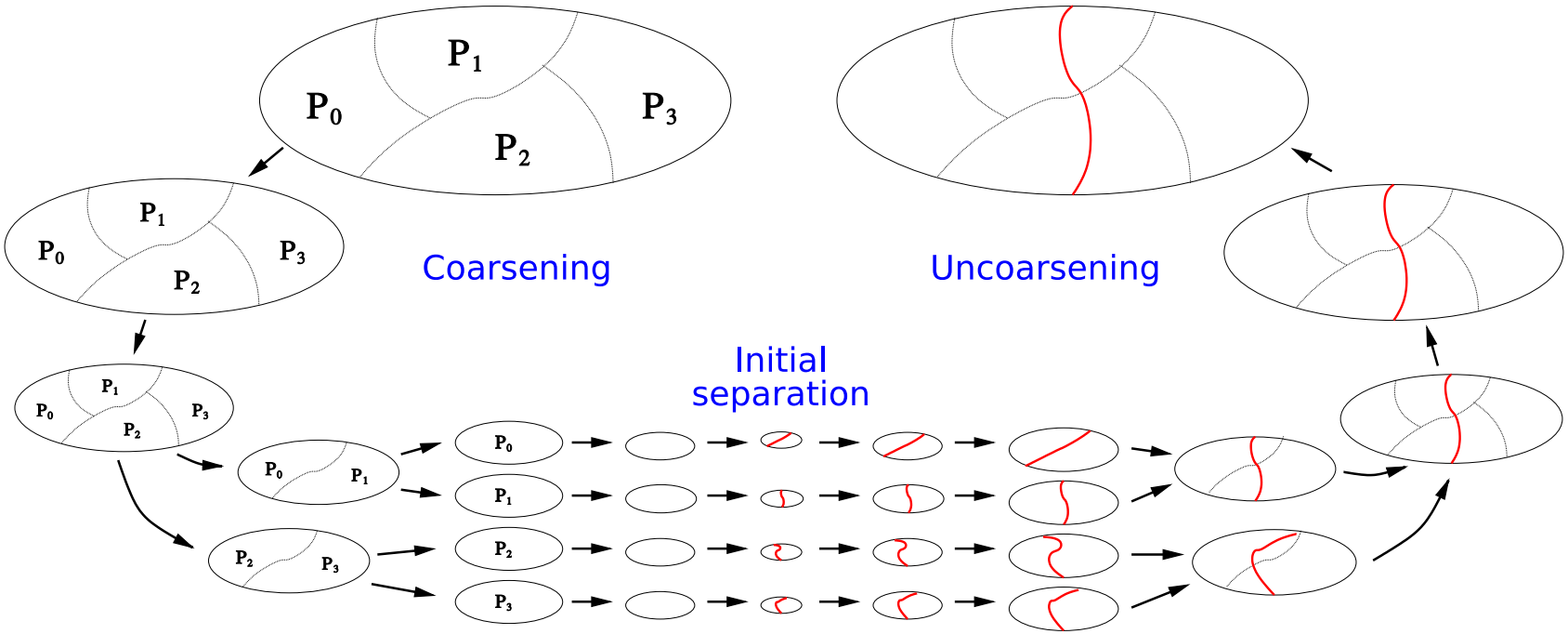
Les sous-renumérotations de A et B sont calculées en parallèle

Parallélisation de l'algorithme Multi-niveaux

- Durant la contraction, chaque sous-graphe contracté est replié et dupliqué sur le sous-ensemble de processeurs restants jusqu'à ce que celui-ci soit réduit à un processeur
- Replier en dupliquant permet d'améliorer la localité des données et d'accélérer l'appariement distribué asynchrone en réduisant le nombre de voisins distants
- Durant l'expansion, seule la meilleure des deux partitions est gardée et projetée sur le niveau plus fin
  $\Longrightarrow$ Améliore la qualité de la partition grâce aux multiples essais

Parallélisation de l'algorithme Multi-niveaux

Schéma du procédé :



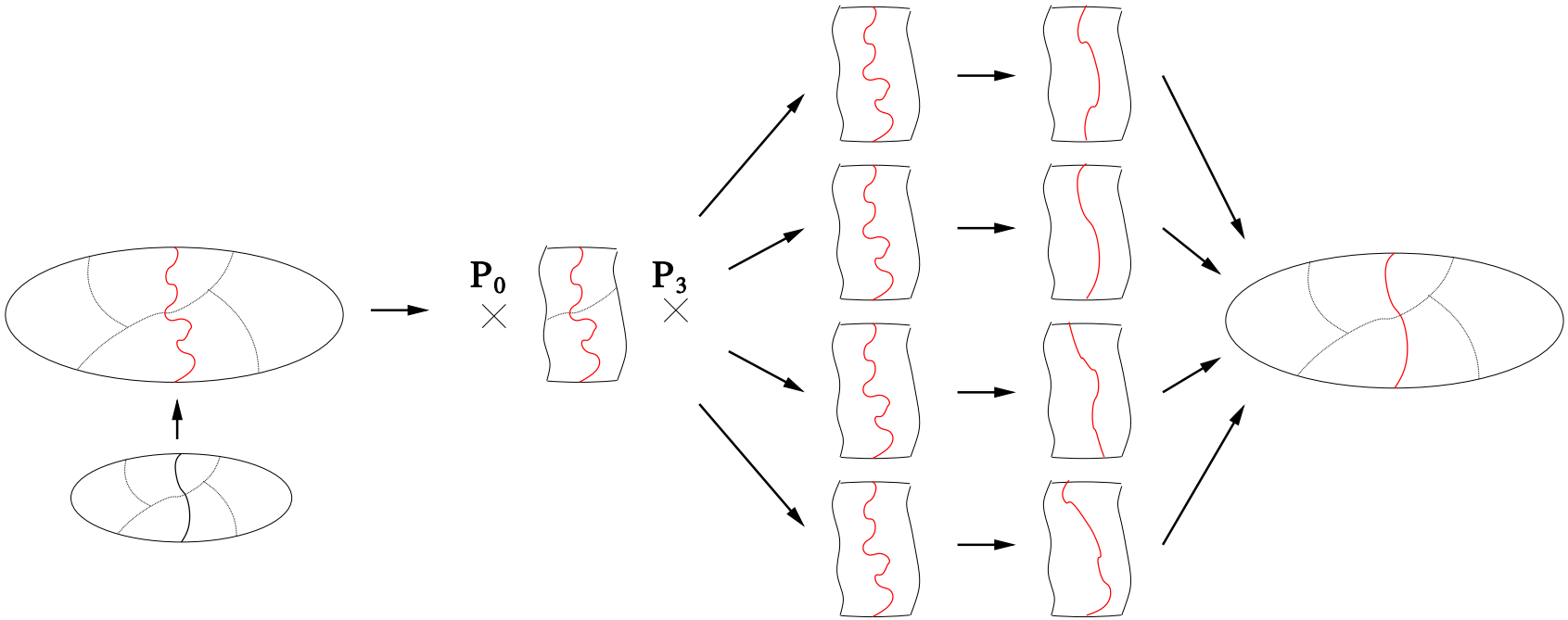
Actuellement, nous utilisons le repliement avec duplication à chaque étape

Parallélisation du raffinement

- Utilisation de la Bande
- Possibilité d'un raffinement multi-séquentiel quand le graphe bande centralisé loge dans la mémoire d'un noeud $\longrightarrow$ Possibilité d'utiliser un F.M. multi-séquentiel pour les graphes 3D jusqu'à un milliard de sommets
- Des algorithmes coûteux mais hautement scalables comme les Algorithmes Génétiques peuvent aussi être utilisés dans les plus hauts niveaux de l'expansion

Parallélisation du raffinement(2)

Schéma du procédé :

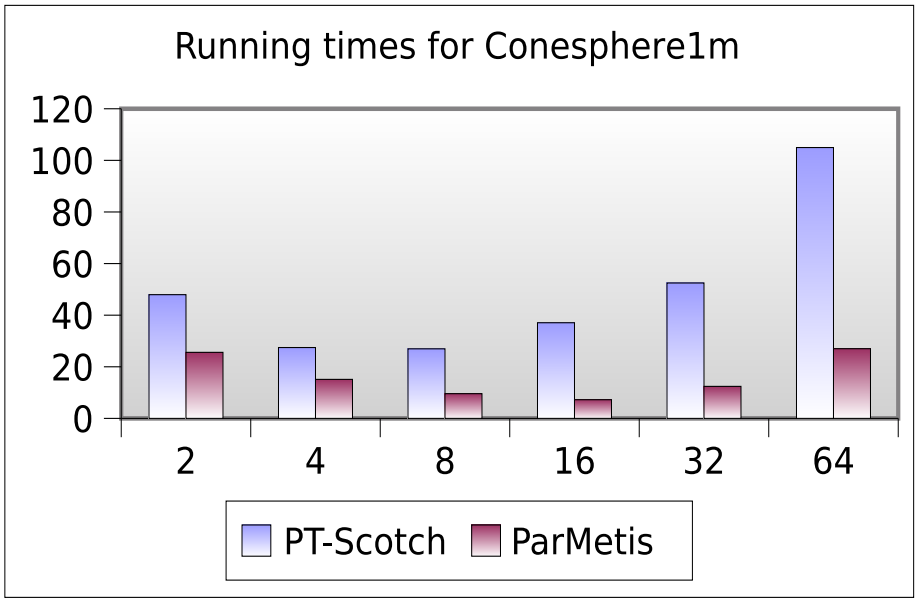


Actuellement nous utilisons un raffinement F.M. multi-séquentiel sur les graphes bandes, sans Algorithmes Génétiques

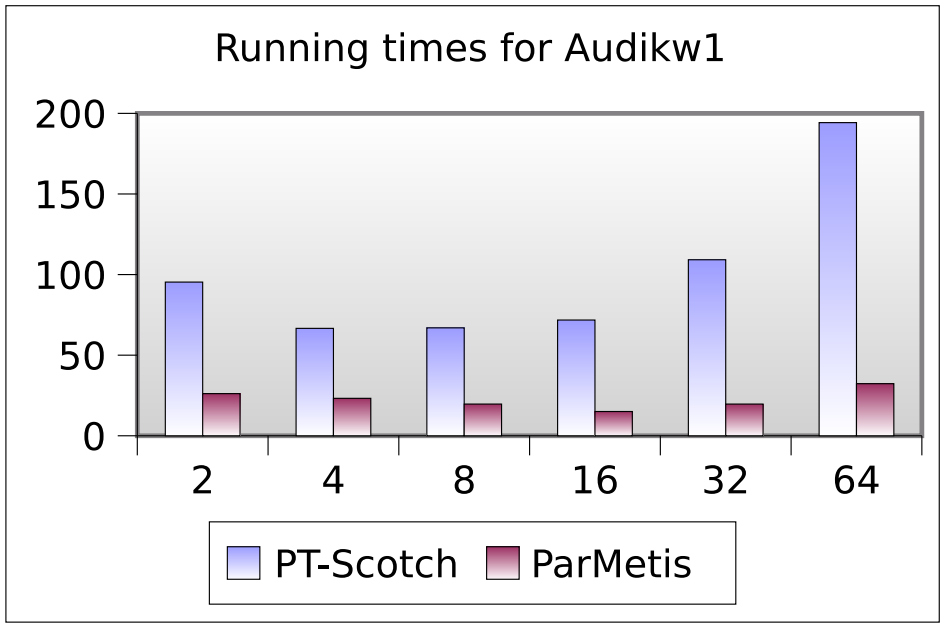
Sommaire

Résultats en temps (1)

Les tests ont été effectués sur une machine octo-processeurs à base d'Opterons bi-cœurs



$|V| = 1.06 \times 10^6$, $|E| = 8.02 \times 10^6$



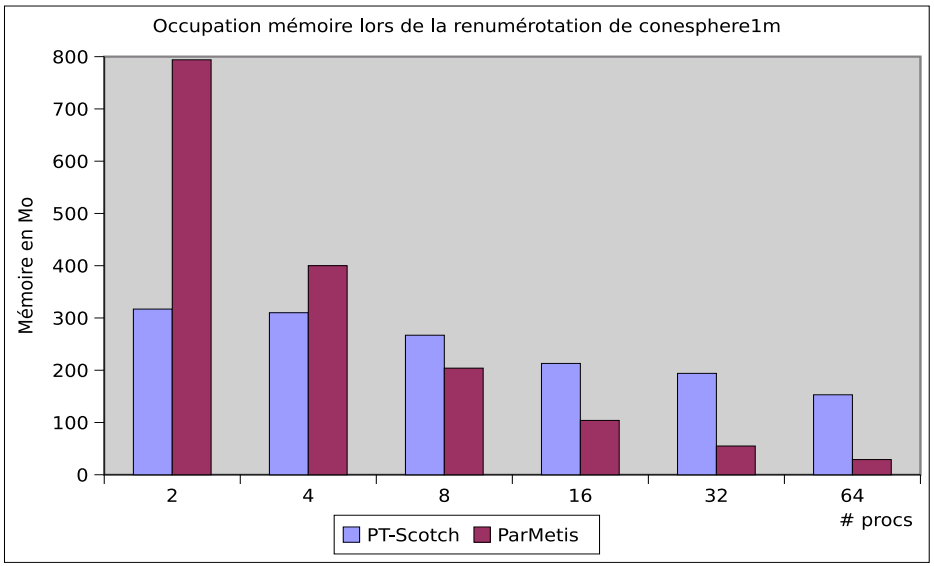
$|V| = 0.94 \times 10^6$, $|E| = 38.35 \times 10^6$

Résultats en temps (2)

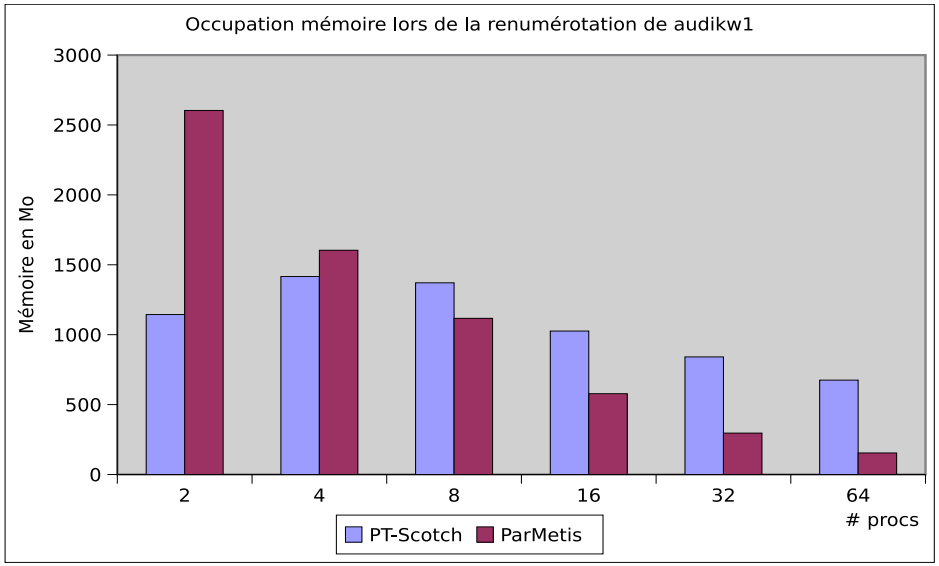
La scalabilité en temps n'est pas bonne quand les graphes n'ont pas assez de sommets, parce que le coût de l'algorithme de contraction augmente considérablement selon le nombre de voisins distants

. . .Mais dans les tests précédents, au dessus de 16 processus, nous séquentialisons les calculs sur les repliements dupliqués qui sont supposés être exécutés en parallèle . . .

Occupation mémoire (1)



$|V| = 1.06 \times 10^6$, $|E| = 8.02 \times 10^6$

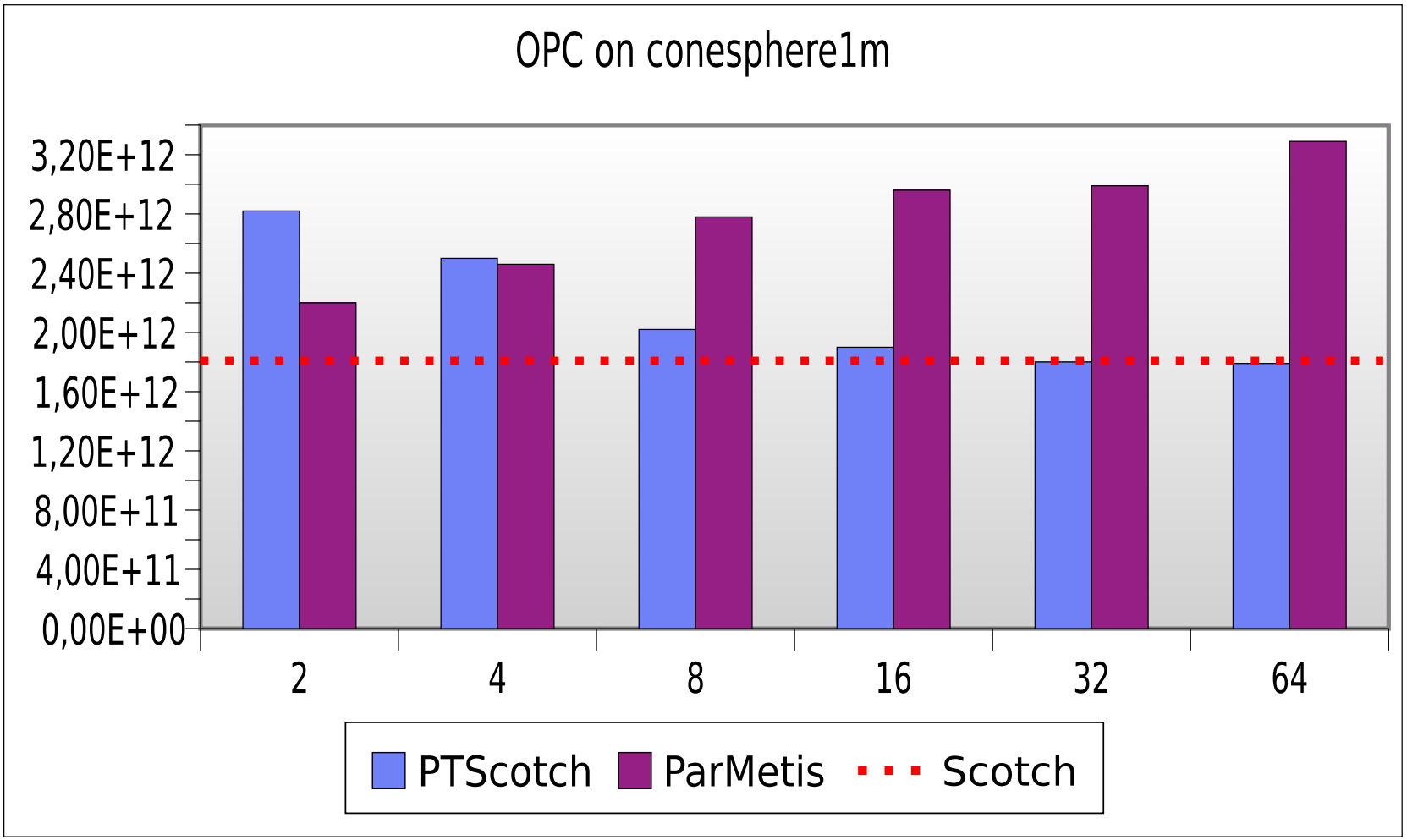


$|V| = 0.94 \times 10^6$, $|E| = 38.35 \times 10^6$

Occupation mémoire (2)

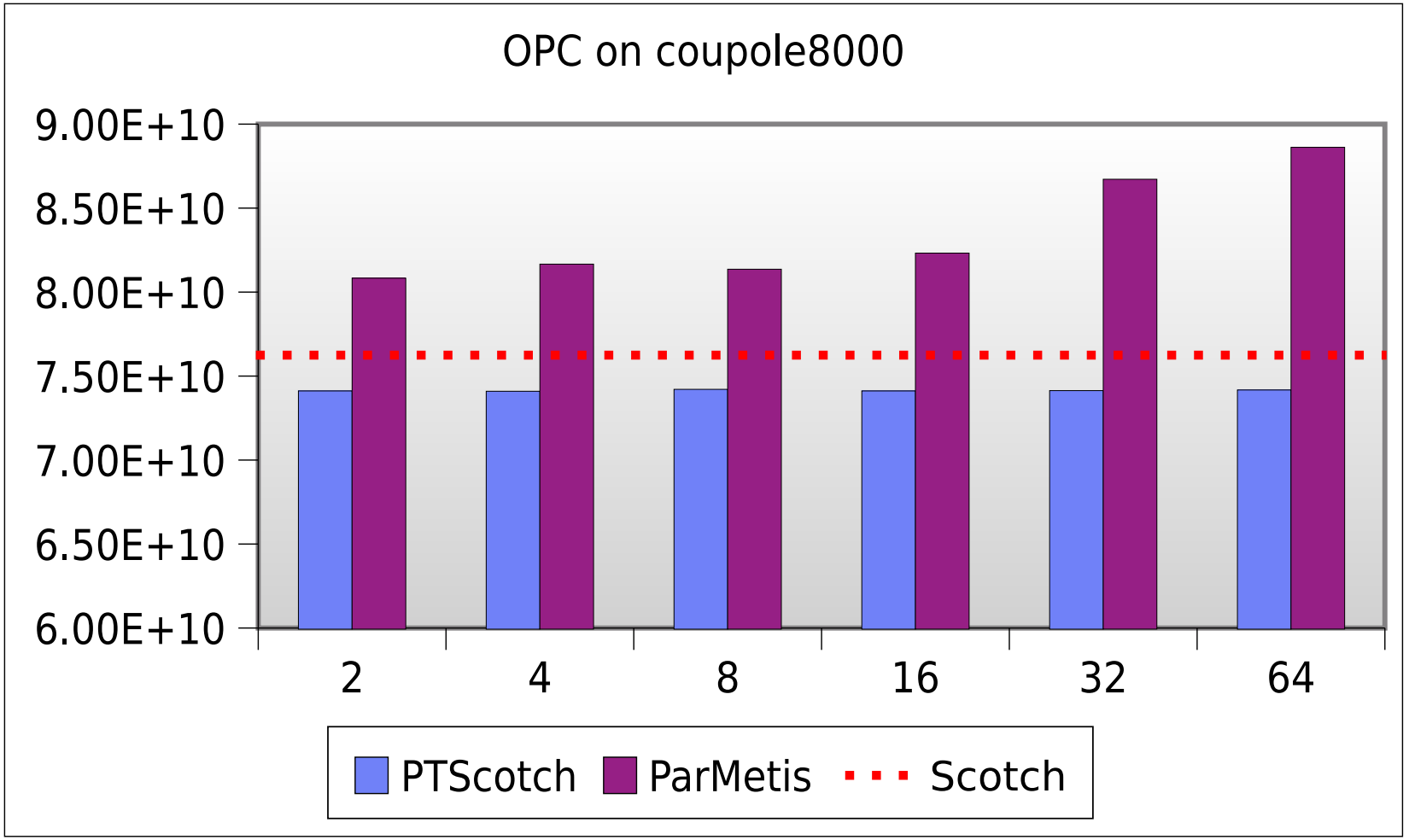
- Avec un faible nombre de processus :
  Occupation mémoire plus faible que PARMETIS
- Quand le nombre de processus augmente :
  Moins scalable que PARMETIS a cause des duplications

Résultats en qualité (1)



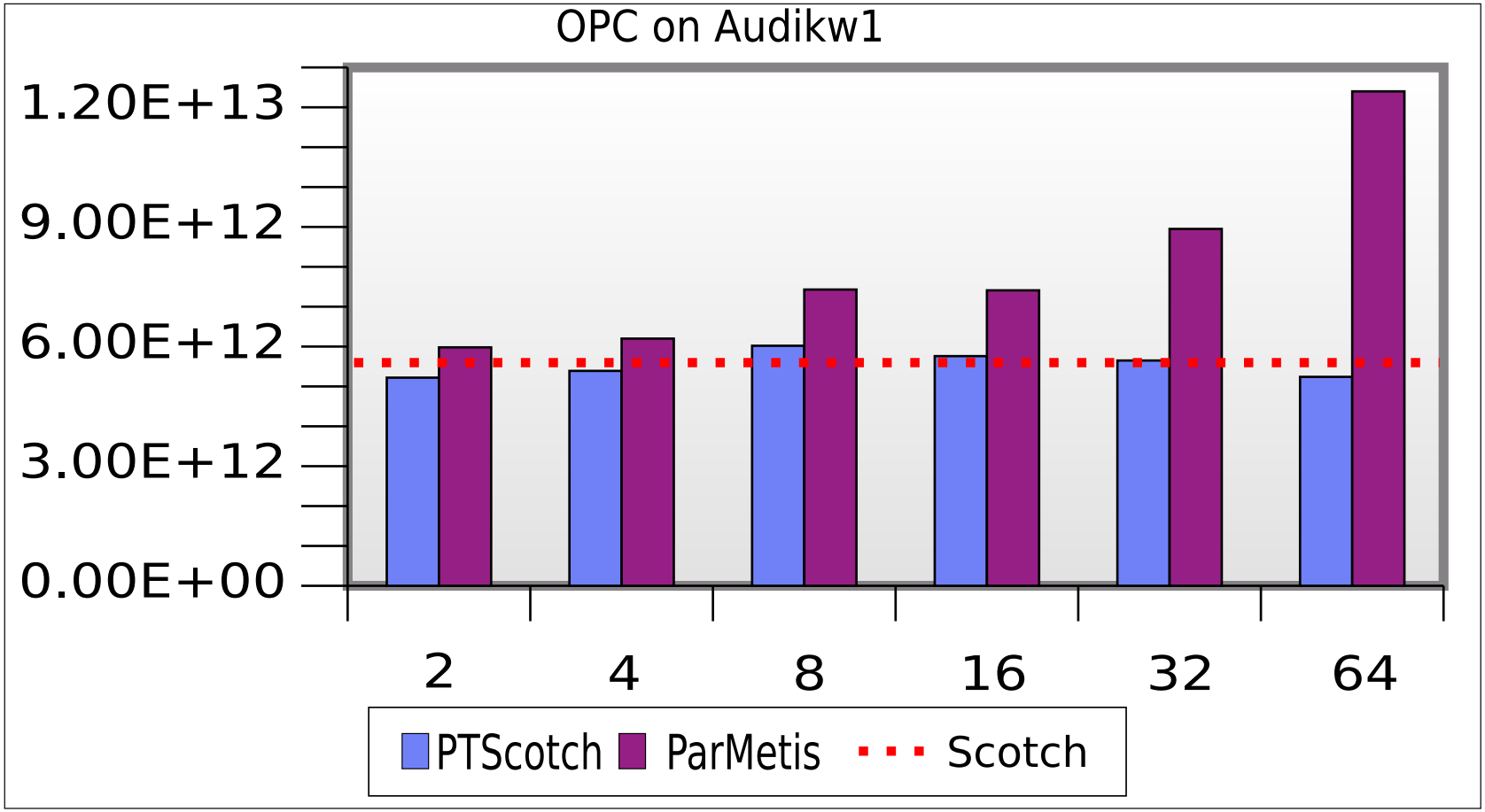
$|V| = 1.06 \times 10^6$, $|E| = 8.02 \times 10^6$

Résultats en qualité (2)



$|V| = 1.77 \times 10^6$, $|E| = 41.66 \times 10^6$

Résultats en qualité (3)



$|V| = 0.94 \times 10^6$, $|E| = 38.35 \times 10^6$

Résultats en qualité (4)

L'amélioration de la qualité avec le nombre de processeurs peut être expliquée par l'augmentation de l'espace de recherche due à la parallélisation dans :

- L'algorithme multi-niveaux : le meilleur séparateur du niveau est choisi parmi les deux calculés par le niveau précédent
- Le raffinement multi-séquentiel : pour chaque paire de graphes fins et contractés, le séparateur du plus fin est le meilleur parmi celui de tous les graphes bandes construits autour de la projection du séparateur du niveau inférieur et raffinés par des F.M. séquentiels

Sommaire

Conclusion

- Une haute qualité de renumérotation parallèle peut être obtenue en parallélisant trois algorithmes clés :
  1. Dissections Emboîtées
  2. Multi-niveaux
  3. Raffinement du séparateur
- Les premiers résultats sont très satisfaisants en terme de qualité mais la scalabilité de l'algorithme d'appariement peut être significativement améliorée

Conclusion

- Une haute qualité de renumérotation parallèle peut être obtenue en parallélisant trois algorithmes clés :
  1. Dissections Emboîtées
  2. Multi-niveaux
  3. Raffinement du séparateur
- Les premiers résultats sont très satisfaisants en terme de qualité mais la scalabilité de l'algorithme d'appariement peut être significativement améliorée

INRIA FUTURS

Travaux en cours

- Optimisation de la contraction en implantant un appariement asynchrone multi-bufferisé (disponible très prochainement)
- Parallélisation du raffinement sur la bande quand les graphes bandes ne tiennent plus en mémoire
  $\longrightarrow$ Utilisation d'un algorithme génétique totalement parallèle
- Conception d'algorithmes efficaces et scalables pour le partitionnement k-aires

Première version publique de PT-SCOTCH (renuméroteur seulement) prévue pour Novembre 2006

`http://www.labri.fr/~pelegrin/scotch`
`http://gforge.inria.fr/projects/scotch`

Merci