TreeMatch: un algorithme de placement de processus sur architectures multicœurs

ComPAS' 2013, Parallélisme - Grenoble

Emmanuel Jeannot Guillaume Mercier François Tessier

20 mai 2013

Contexte

- Applications hautes performances
- Architectures multi-noeuds et multi-coeurs

Interface à Passage de Messages (MPI)

- Bibliothèque standardisée
- Communications entre processus (affinités)
- Pas d'hypothèses sur le placement des processus
- Quelles performances?

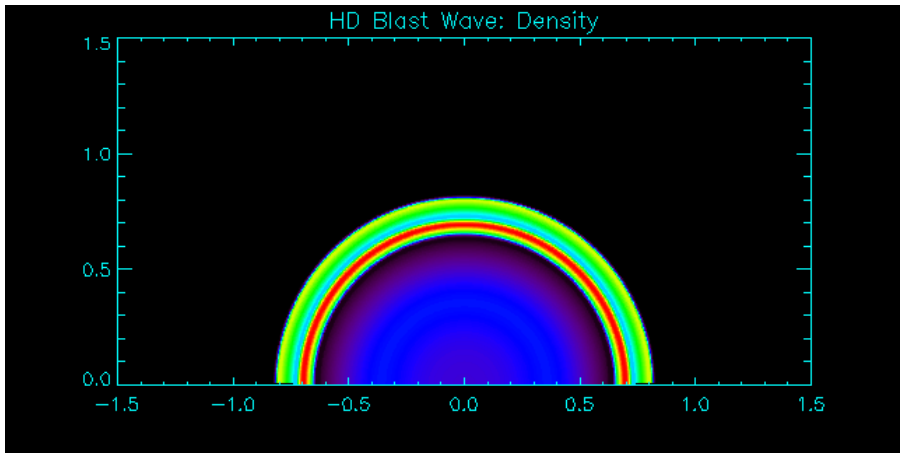


Figure : Cas *HDBlast* de l'application ZeusMP/2

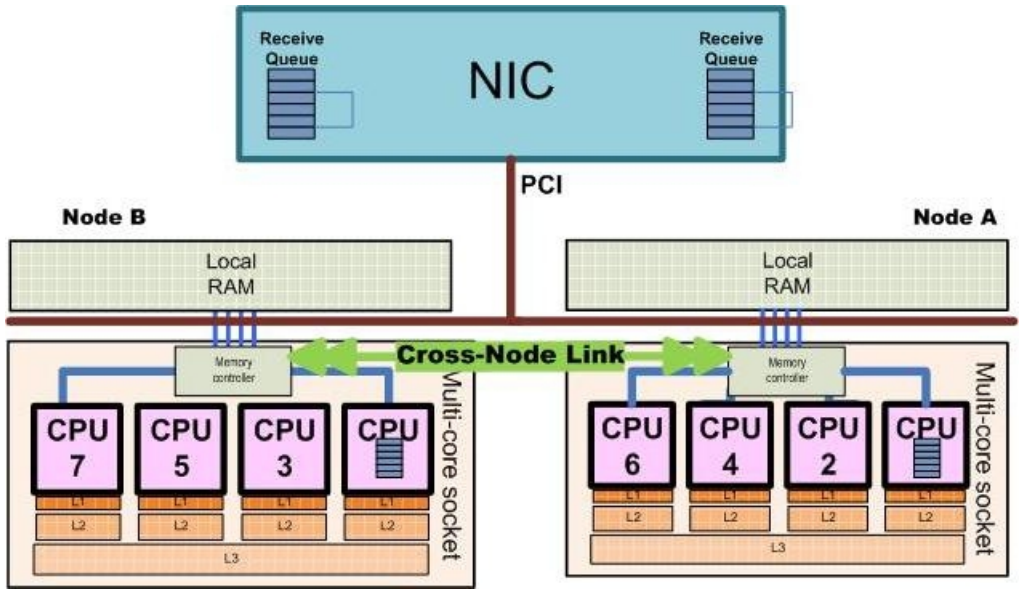
Problématique

Logiciel

- Implémentation de MPI
- Structure de l'application

Matériel

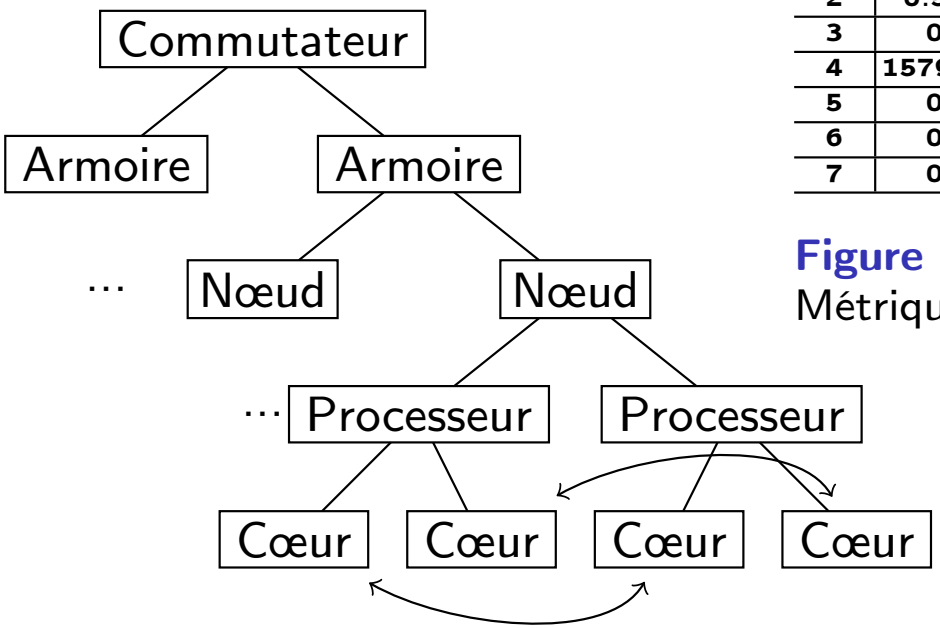
- Architectures fortement hierarchiques
  - Réseau;
  - Caches;
  - Coeurs...
- Effets NUMA



Comment minimiser les coûts de communication entre processus?

Pourquoi?

- Volumes de données différents échangés entre processus (affinités)
- Vitesses de communication (Réseau, bus mémoire, ...)
- Performances impactées par le placement des processus



Proc	0	1	2	3	4	5	6	7
0	0	311811	0.56	0	157979	0	0	0
1	311811	0	311810	0.56	0	159986	0	0
2	0.56	311810	0	311811	0	0	153832	0
3	0	0.56	311811	0	0	0	0	151825
4	157979	0	0	0	0	311811	0.48	0
5	0	159986	0	0	311811	0	311810	0.48
6	0	0	153832	0	0.48	311810	0	311811
7	0	0	0	151825	0	0.48	311811	0

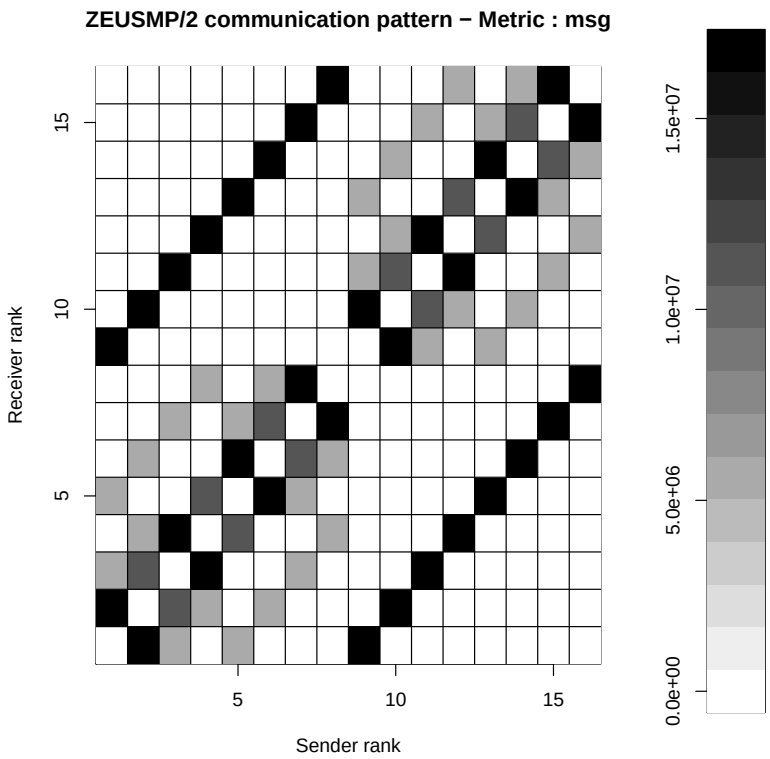
Figure : Matrice de communication de lu.C.8 (NAS). Métrique de volume de données échangé, en ko

Formalisation

Soit :

- Le modèle de communication de l'application
- Une représentation de l'architecture

Objectif : Trouver une permutation des processus de façon à réduire les coûts de communication.

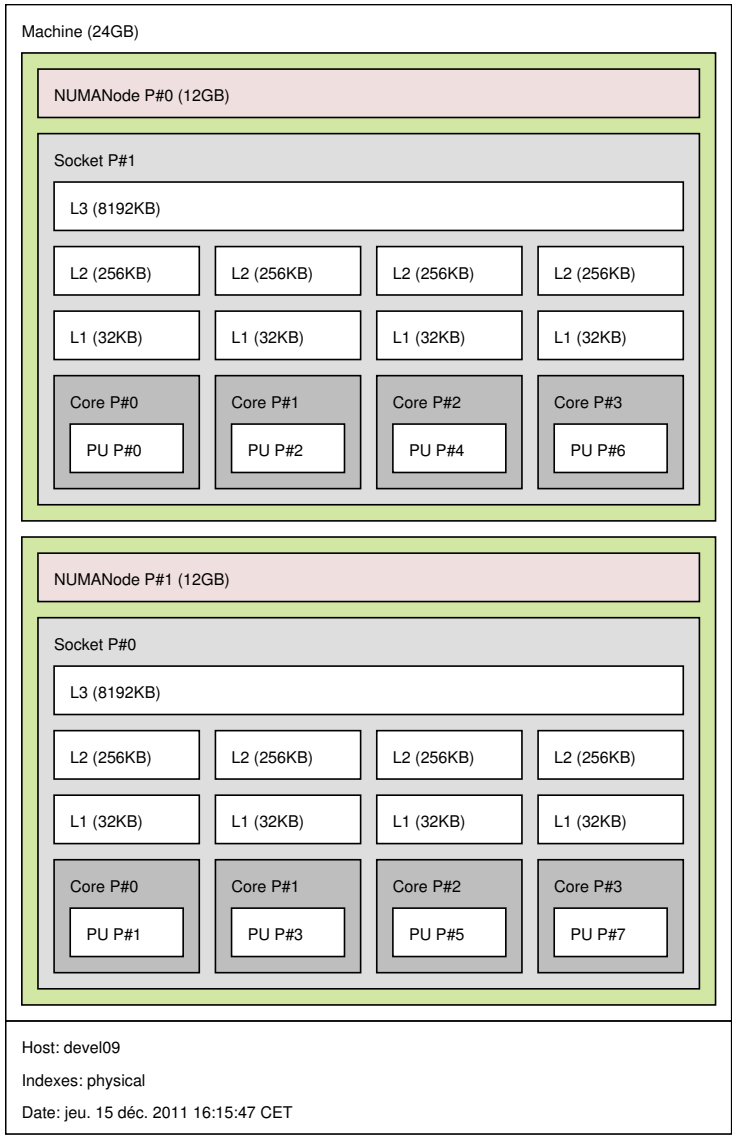


Modèle de communication

- Matrice de communication de taille $p \times p$, avec p le nombre de processus de l'application
- Métriques
  - size : volume de données
  - msg : nombre de messages
  - avg : taille moyenne d'un message
- Modification des implémentations MPI pour monitorer les communications (point à point, collectives)

Topologie

- Hwloc : bibliothèque développée par Inria et l'équipe OpenMPI
- Représentation précise de l'architecture matérielle
- Portable, multi-plateforme
- Architectures modernes (NUMA, cœurs, caches, ...)
- Structure arborescente

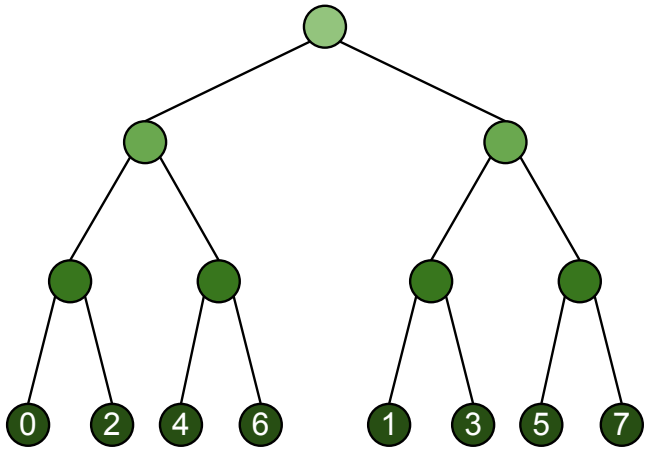


Principe

- Exécutable et bibliothèque
- Calcul d'une permutation de processus
- Fonction des affinités entre processus et de l'architecture cible
- Soit
  - p : le nombre de processus
  - n : le nombre d'unités de calcul (feuilles de l'arbre), $n \geq p$
  - σ_i : l'unité de calcul qui exécute le processus i. $\sigma_i \in [1, n], i \in [1, p]$

Fonctionnement par l'exemple

Topologie



Matrice de communication

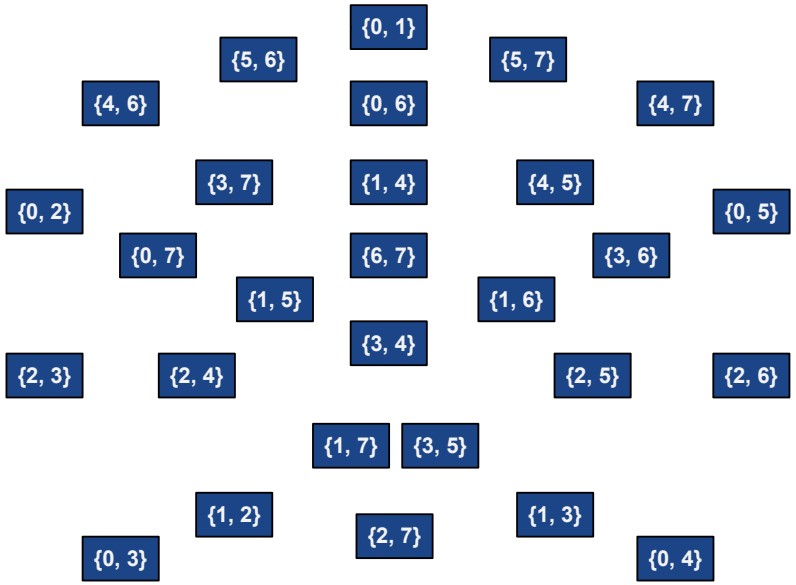
Proc	0	1	2	3	4	5	6	7
0	0	100	10	1	1000	1	1	1
1	100	0	100	1	1	1000	1	1
2	10	100	0	100	1	1	1000	1
3	1	1	100	0	1	1	1	1000
4	1000	1	1	1	0	100	10	1
5	1	1000	1	1	100	0	100	1
6	1	1	1000	1	10	100	0	100
7	1	1	1	1000	1	1	100	0

Algorithme

- Topologie à 4 niveaux de profondeur. Arité du niveau 2 : $k = 2$

Fonctionnement par l'exemple

Groupes de processus



Matrice de communication

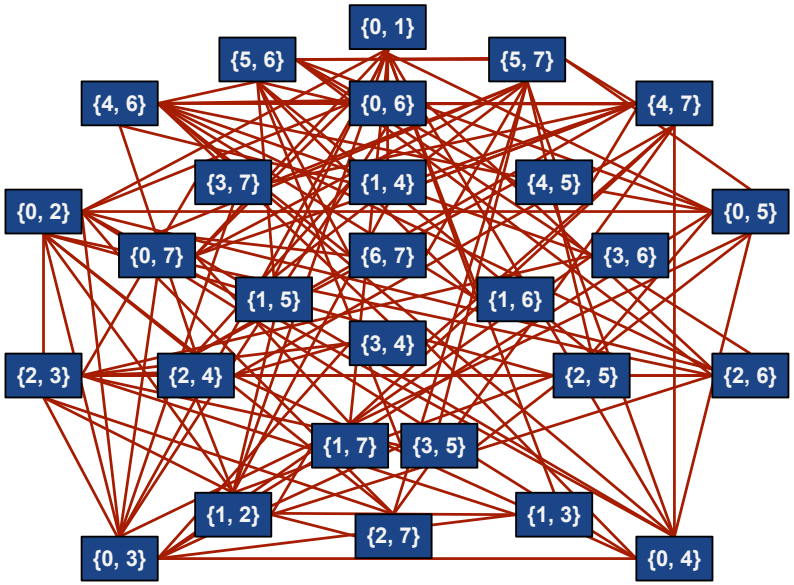
Proc	0	1	2	3	4	5	6	7
0	0	100	10	1	1000	1	1	1
1	100	0	100	1	1	1000	1	1
2	10	100	0	100	1	1	1000	1
3	1	1	100	0	1	1	1	1000
4	1000	1	1	1	0	100	10	1
5	1	1000	1	1	100	0	100	1
6	1	1	1000	1	10	100	0	100
7	1	1	1	1000	1	1	100	0

Algorithme

- Topologie à 4 niveaux de profondeur. Arité du niveau 2 : $k = 2$
- On sélectionne 4 groupes de k processus
  - $\binom{8}{2} = 28$ groupes possibles

Fonctionnement par l'exemple

Graphe des incompatibilités



Matrice de communication

Proc	0	1	2	3	4	5	6	7
0	0	100	10	1	1000	1	1	1
1	100	0	100	1	1	1000	1	1
2	10	100	0	100	1	1	1000	1
3	1	1	100	0	1	1	1	1000
4	1000	1	1	1	0	100	10	1
5	1	1000	1	1	100	0	100	1
6	1	1	1000	1	10	100	0	100
7	1	1	1	1000	1	1	100	0

Exemple avec le groupe $\{0, 1\}$: **2118**

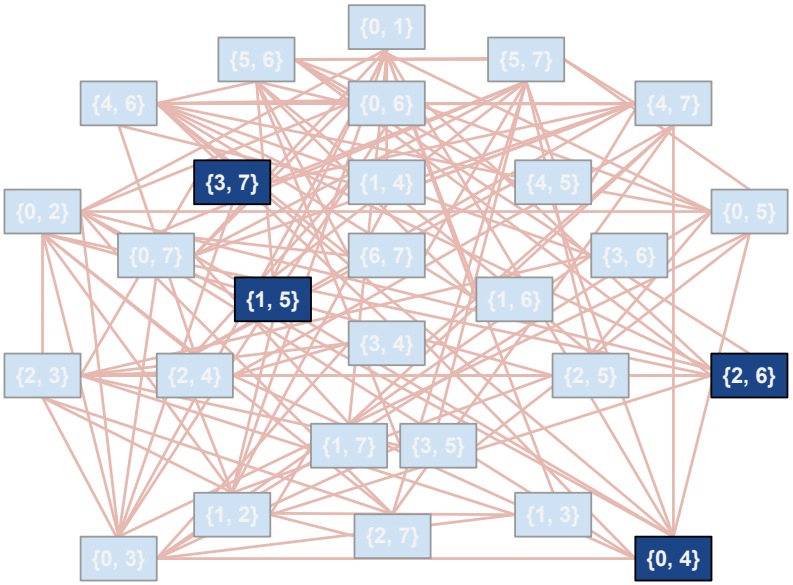
$$\sum_{i=0}^{p} m[0, i] + \sum_{i=0}^{p} m[1, i] - (m[0, 1] + m[1, 0])$$

Algorithme

- Topologie à 4 niveaux de profondeur. Arité du niveau 2 : $k = 2$
- On sélectionne 4 groupes de k processus
  - $\binom{8}{2} = 28$ groupes possibles
  - Graphe des incompatibilités
  - Pondération des groupes avec le volume de communication restant

Fonctionnement par l'exemple

Ensemble indépendant



Matrice de communication

Proc	0	1	2	3	4	5	6	7
0	0	100	10	1	1000	1	1	1
1	100	0	100	1	1	1000	1	1
2	10	100	0	100	1	1	1000	1
3	1	1	100	0	1	1	1	1000
4	1000	1	1	1	0	100	10	1
5	1	1000	1	1	100	0	100	1
6	1	1	1000	1	10	100	0	100
7	1	1	1	1000	1	1	100	0

Algorithme

- Topologie à 4 niveaux de profondeur. Arité du niveau 2 : $k = 2$
- On sélectionne 4 groupes de k processus
  - $\binom{8}{2} = 28$ groupes possibles
  - Graphe des incompatibilités
  - Pondération des groupes avec le volume de communication restant
  - Heuristique afin de trouver un «bon» ensemble indépendant de poids minimal

Fonctionnement par l'exemple

Matrice agrégée (prof. 2)

Virt. Proc	0	1	2	3
0	0	202	22	4
1	202	0	202	4
2	22	202	0	202
3	4	4	202	0

Matrice agrégée (prof. 1)

Virt. Proc	0	1
0	0	232
1	232	0

Suite

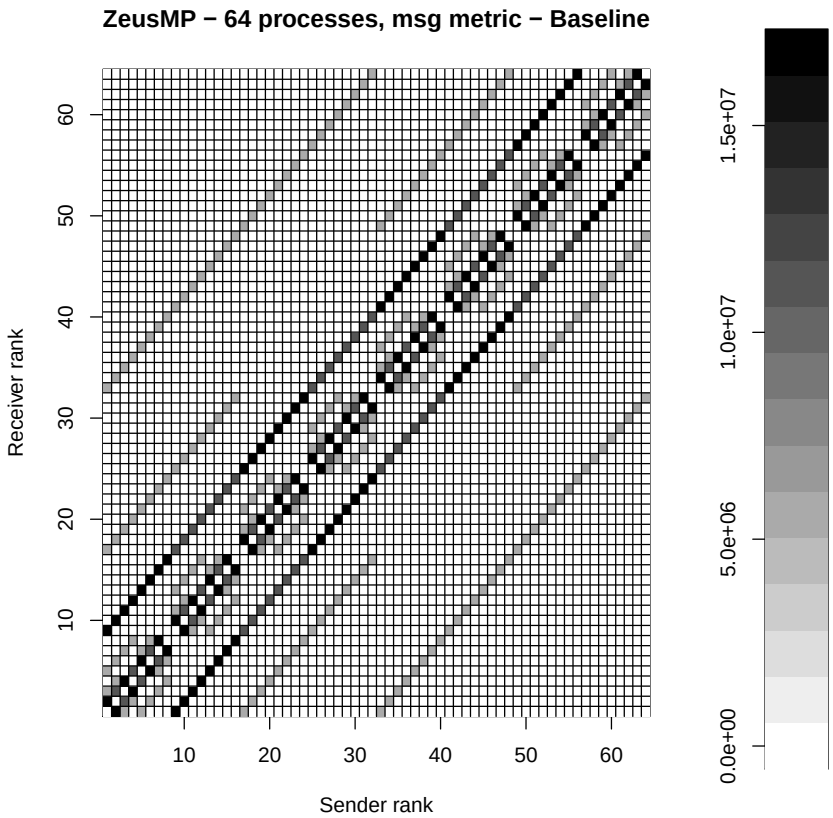
- Nouvelle matrice de communication entre processus virtuels
- Même procédure sur les matrices «virtuelles» et sur le sous-arbre correspondant
- Hierarchie de groupes qui va déterminer le placement

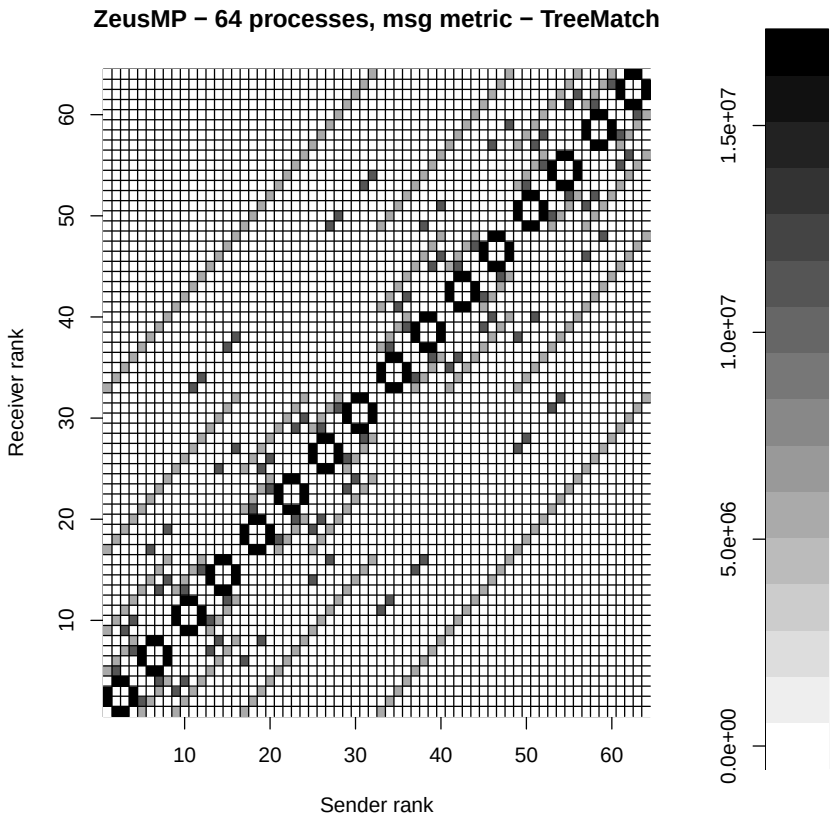
Permutations calculées

- TreeMatch : **0,4,1,5,2,6,3,7**
- Packed (identité logique) : **0,2,4,6,1,3,5,7**
- Round Robin (identité physique) : **0,1,2,3,4,5,6,7**

Cas ZeusMP/2

- Application de mécanique numérique des fluides
- Beaucoup de communications
- Cas sur 64 processus





Autres méthodes de placement

Partitionneurs de graphes et algorithmes randomisés

- Chaco [Hendrickson & Leland, 1994]: Partitionneur de graphes. Bipartitions successives.
- MPIPP [Chen et al., 2006] : Algorithme randomisé de calcul de permutations.
- Scotch [Pellegrini, 1994] : Partitionneur de graphes, outils de calcul de placement de processus sur architecture arborescente.
- (Par)METIS [Karypis & Kumar, 1995] : Partitionneur de graphe, bipartitions successives.

Scotch

- Arbre de la topologie pondéré
- Contraintes fortes sur les poids

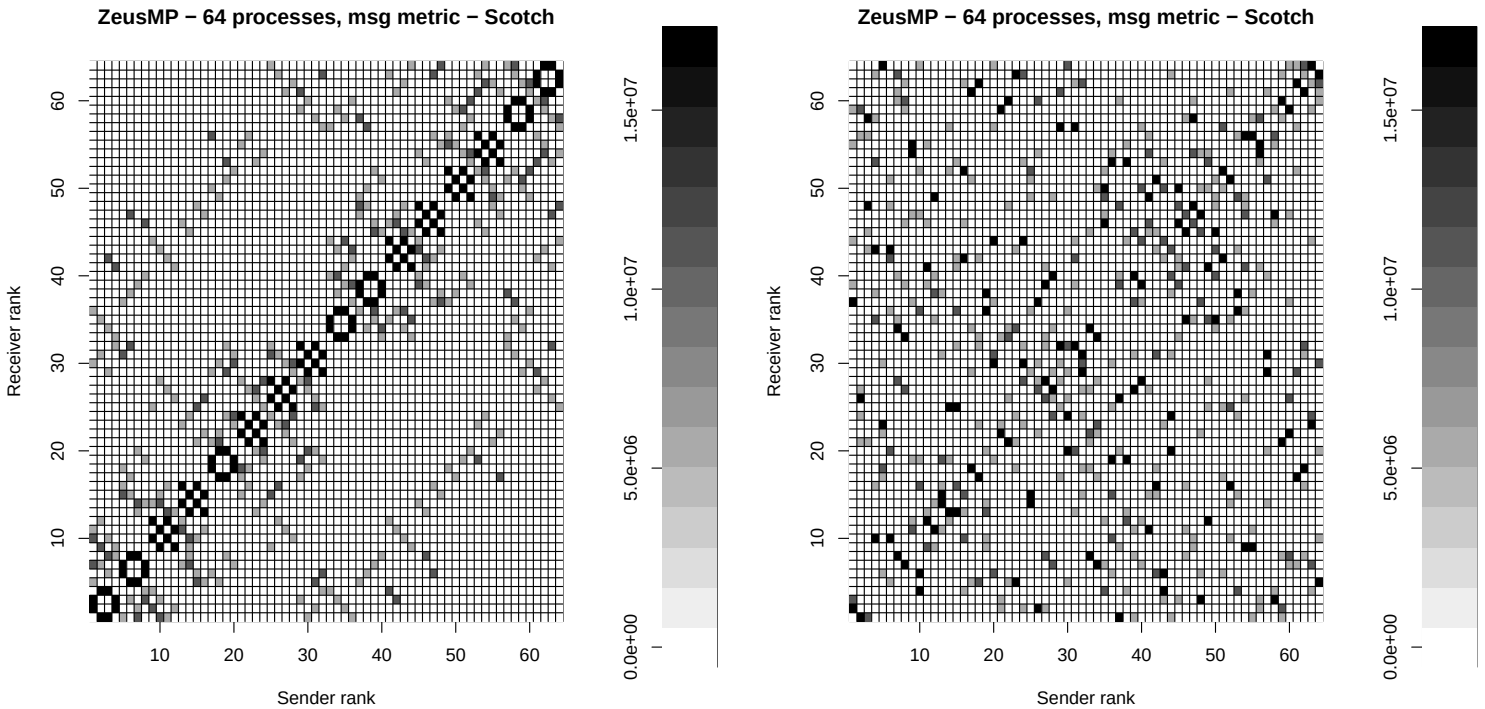


Figure : Permutations calculées par Scotch avec des poids faibles (< 10) et des poids plus importants (< 200)

TreeMatch

- Indépendant de toute valuation
- Basé sur la distance dans l'arbre de la topologie

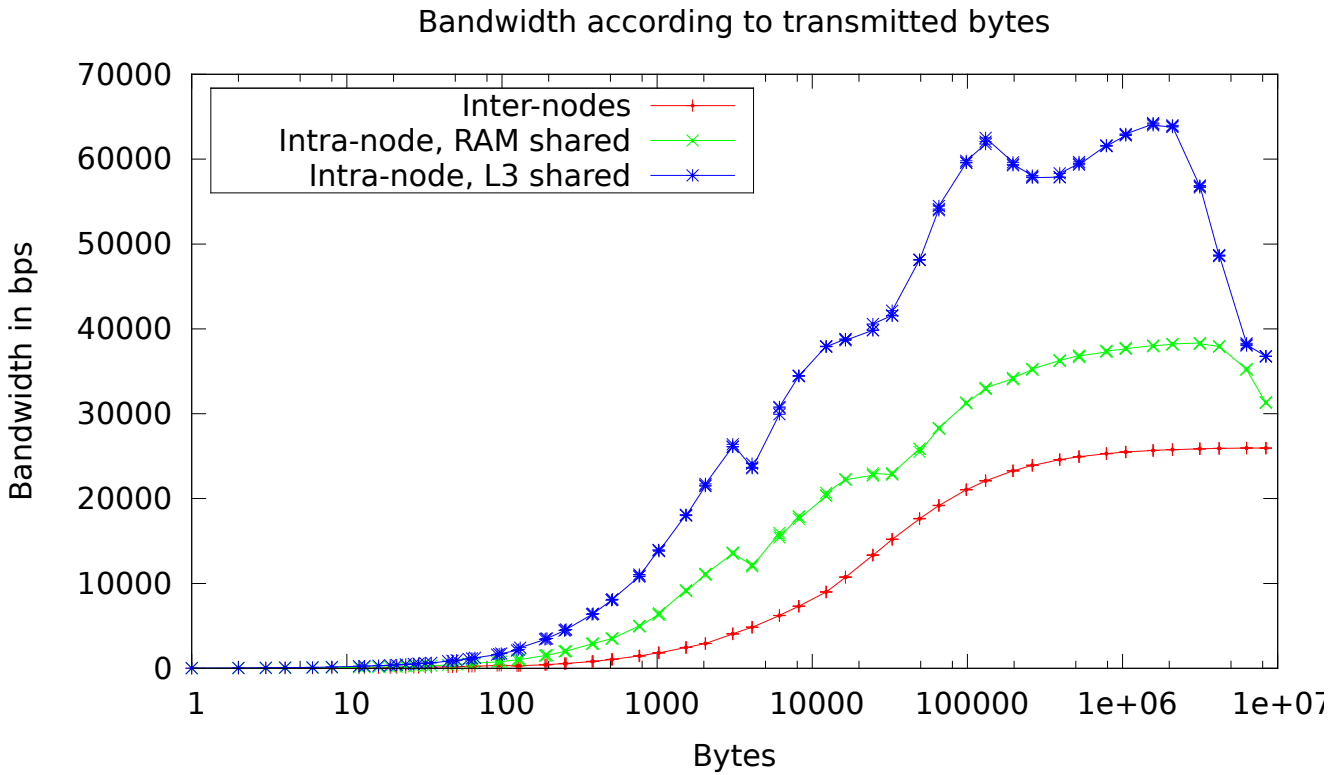


Figure : Bande passante en fonction du nombre d'octets échangés. Placement de 2 processus MPI, 1000 mesures pour chaque point.

Résultats

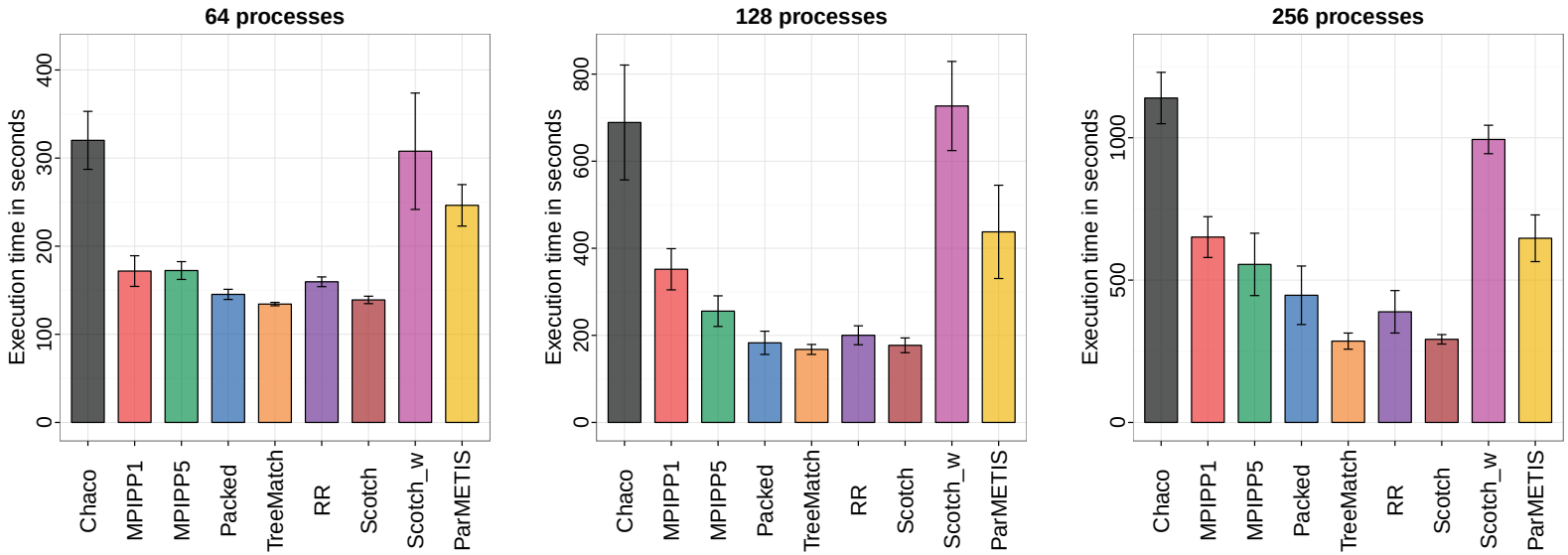


Figure : Temps moyens d'exécution de ZeusMP/2 en faisant varier le nombre de processus (moyennes sur 10 exécutions entrelacées, metrique *msg*, 3000 itérations)

Interprétation

- Gain de 30% par rapport à Round Robin, placement par défaut de MPI, sur 256 processus
- Différences importantes face à Chaco ou ParMETIS (bipartitions successives)
- Scotch dépendant de la valutation de l'arbre de la topologie

Conclusion

Bilan

- Architectures hierarchiques et affinité des processus non uniforme
- Traitement des applications sans les modifier
- Approche basée sur des informations quantitatives et non qualitatives
- Performances accrues par rapport au placement par défaut de MPI

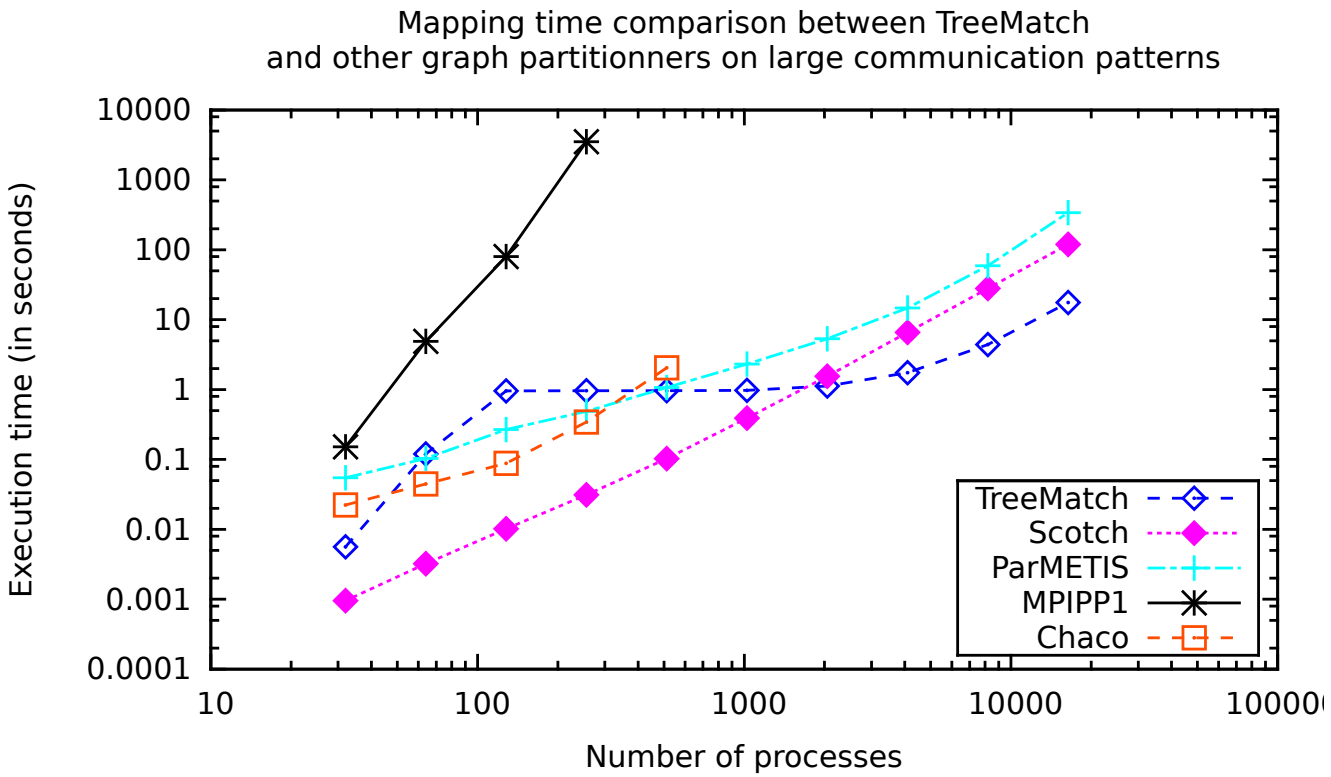
Perspectives

- Étendre à plus d'architectures (tore, grille)
- Collecte du modèle de communication via de l'analyse statique de code
- Équilibrage de charge
- Placement dynamique

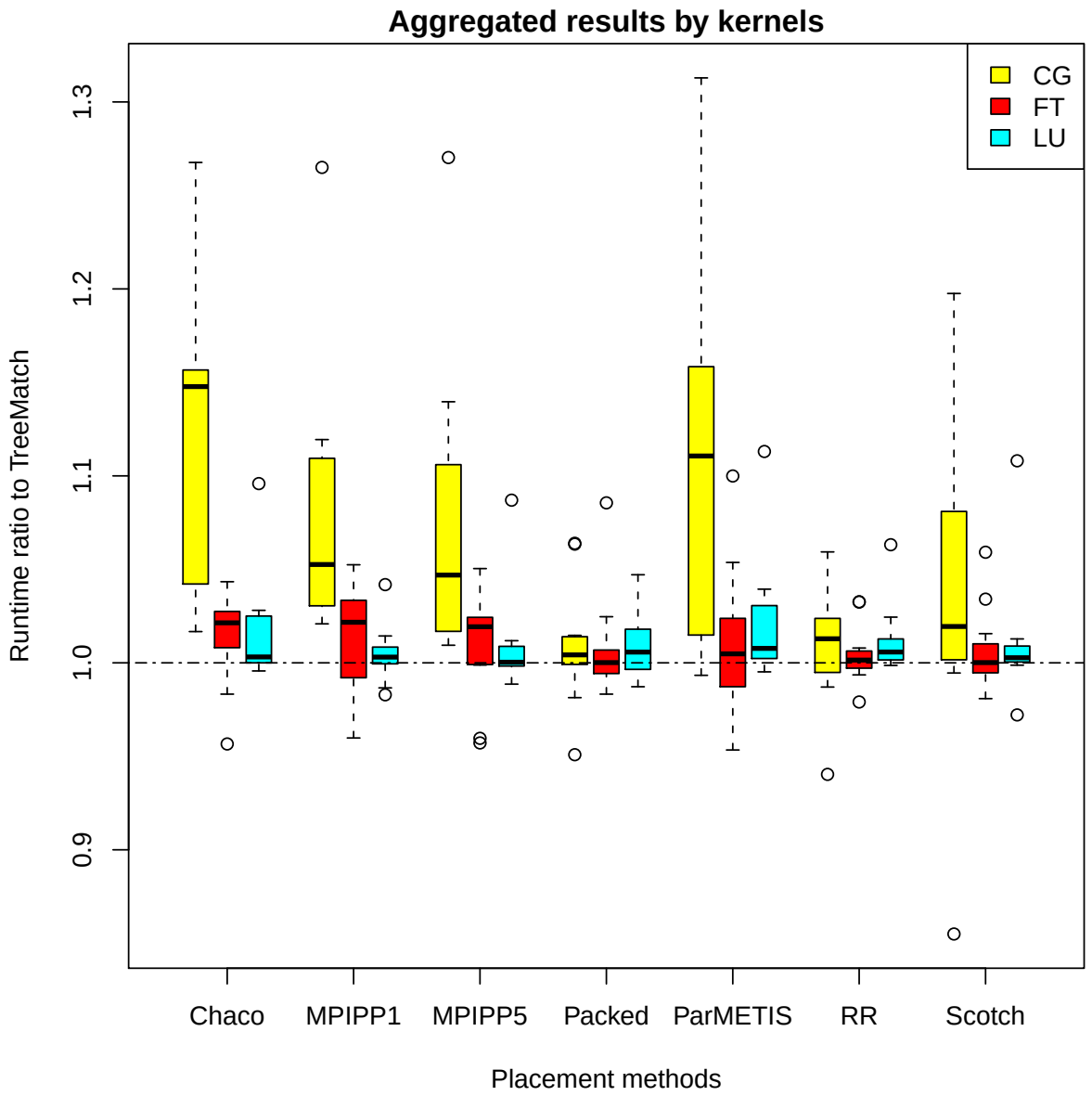
Merci de votre attention !

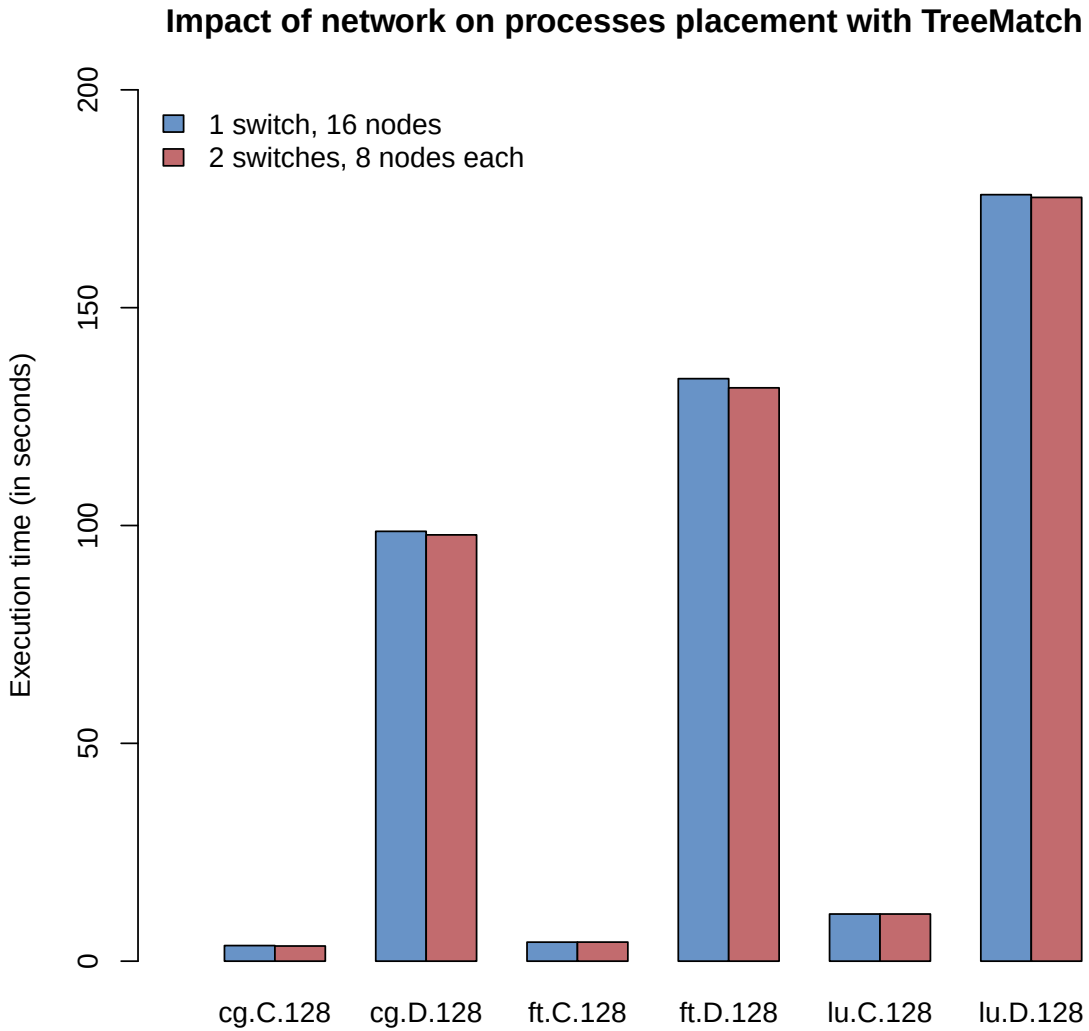
Des questions ?

Annexes - Temps de calcul des permutations



Annexes - NAS Parallel Benchmarks




Impact of network on processes placement with TreeMatch
1 switch, 16 nodes
2 switches, 8 nodes each
Execution time (in seconds)
0
50
100
150
200
cg.C.128
cg.D.128
ft.C.128
ft.D.128
lu.C.128
lu.D.128

Annexes - Bipartitions successives