# SMPI : Simulation d'applications MPI dans SimGrid

Pierre-Nicolas Clauss

Université de Nancy, Équipe AlGorille

17 février 2011

# Plan

- Introduction
- Précision
- Scalabilité
- Vitesse
- Conclusion et perspectives

# Plan

# Motivations : pourquoi utiliser la simulation ?

- Prédiction de performance (hypothèses scénaristiques)
  - Spécification de plateforme
  - Adaptation matérielle
  - Paramétrage applicatif
- Apprentissage (programmation parallèle, calcul haute-performance)
  - Pas de recours nécessaire à des plateformes réelles
  - Environnement local, pas d'interférences
- Défis à relever
  - Précision : la simulation reflète-t-elle la réalité ?
  - Scalabilité : quelle taille d'application simuler ? sur quelles plateformes ?
  - Vitesse : la simulation est-elle plus rapide que l'exécution réelle ?
  - Reproductibilité : les résultats sont-ils stables et réutilisables ?

# Approches : comment faire une simulation ?

- Simulation *off-line*
  - Acquisition et rejeu de traces
  - Liée à un jeu de paramètres applicatifs
  - Besoin d'une taille minimale de plateforme
  - Impossibilité de simuler des applications sensibles au réseau
  - LogGOPSim, PSiNS
- Simulation *on-line*
  - Exécution directe du code
  - Simulation des temps de communication
  - Une seule machine nécessaire
  - MPI-NetSim, SMPI

# Simulation *on-line* dans SMPI

- Implémentation partielle de MPI au-dessus de SimGrid
- Pas ou peu de modification du code (C ou Fortran)
- Exécution réelle du code sur une machine hôte
  - Report des temps d'exécution dans la simulation
  - Modèle CPU : ratio de puissance
- Simulation des communications
  - Modèle réseau par flux (ethernet)
  - Validité des modèles pour les applications MPI
- Repliement d'une application répartie dans un seul processus système
  - Sérialisation des calculs
  - Espace d'adressage unique
  - Nécessité de réduire l'emprunte mémoire (scalabilité) et CPU (vitesse)

# Modèles réseaux existants dans SimGrid

- Caractéristiques des liens : latence (L) et bande-passante (B)
- Simulation par flux
  - Simulation rapide
  - Calcul de contention simple
- Modèle simple : $T(S) = L + \frac{S}{B}$
  - Modèle valide pour $S \geq 10$ Mo
- Modèle amélioré : $T(S) = \alpha \cdot L + \frac{S}{min\left(\beta \cdot B, \frac{\gamma}{2 \cdot L}\right)}$
  - $\alpha$ permet de modéliser le slow-start de TCP
  - $\beta$ permet de modéliser le surcoût des en-têtes TCP/IP
  - $\gamma$ permet de modéliser les effets de la fenêtre TCP
  - Modèle valide pour $S \geq 100$ Ko, trop gros pour des applications MPI
  - Besoin d'un nouveau modèle réseau

# Validation expérimentale : mesure de l'erreur

- Comparer un temps réel $R$ et un temps simulé $S$
- Différence : $Err = S - R$

  - $R = 10$ ms
  - $S = 9$ ms
  - $Err = -1$ ms

  - $R = 1$ s
  - $S = 1.001$ s
  - $Err = 1$ ms

# Validation expérimentale : mesure de l'erreur

- Comparer un temps réel $R$ et un temps simulé $S$
- Erreur relative : $Err = \frac{S-R}{R}$

  - $R = 10$ ms
  - $S = 9$ ms
  - $Err = -0.1$ (-10 %)

  - $R = 1$ s
  - $S = 1.001$ s
  - $Err = 0.001$ (0.1 %)

# Validation expérimentale : mesure de l'erreur

- Comparer un temps réel $R$ et un temps simulé $S$
- Erreur relative : $Err = \frac{S-R}{R}$

  - $R = 1$ s
  - $S = 0.5$ s
  - $Err = -0.5$ (-50 %)

  - $R = 1$ s
  - $S = 2$ s
  - $Err = 1$ (100 %)

# Validation expérimentale : mesure de l'erreur

- Comparer un temps réel $R$ et un temps simulé $S$
- Erreur logarithme : $Err = |ln(S) - ln(R)| = |ln(R) - ln(S)|$

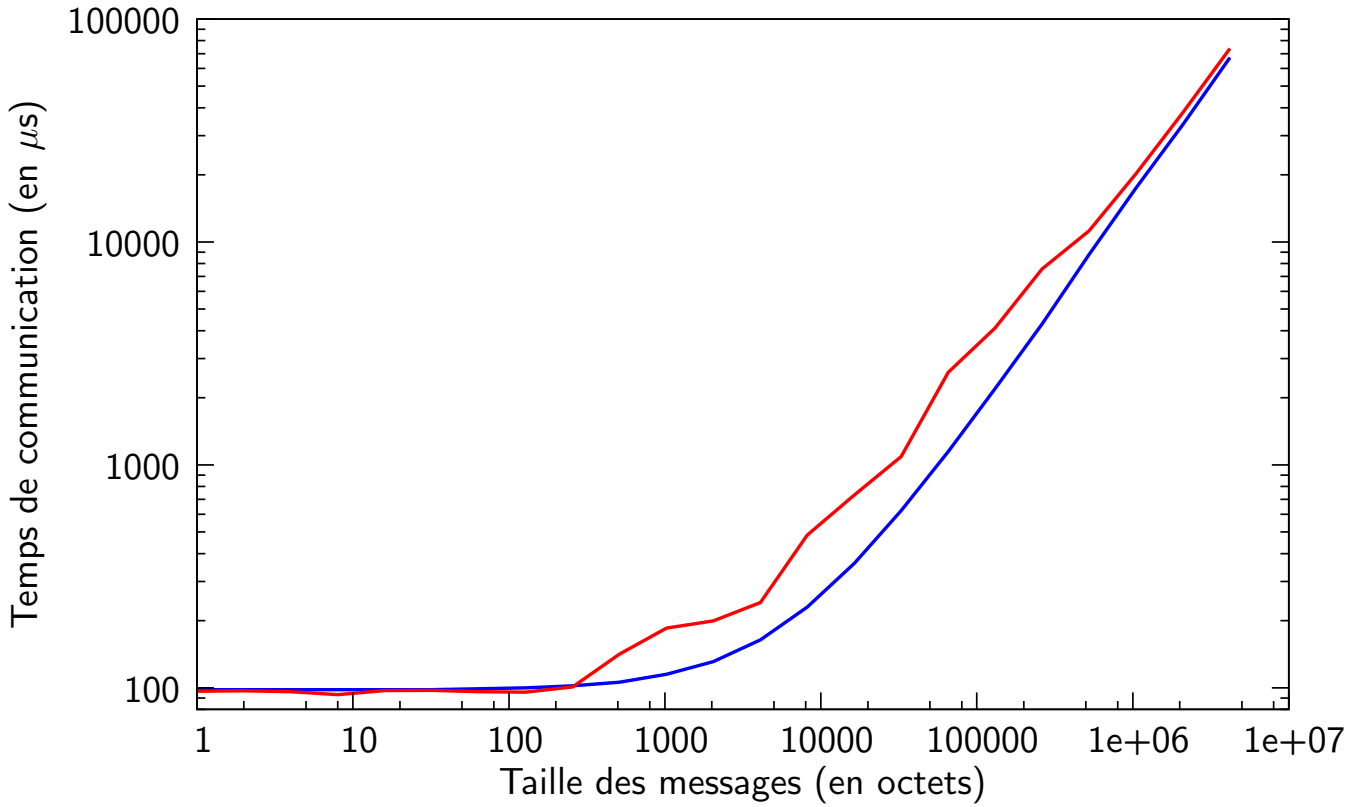
  - $R = 1$ s
  - $S = 0.5$ s
  - $Err = 0.69$ (100 %)

  - $R = 1$ s
  - $S = 2$ s
  - $Err = 0.69$ (100 %)

# Plan

- Introduction
- Précision
  - Opérations point-à-point
  - Opérations collectives
- Scalabilité
- Vitesse
- Conclusion et perspectives

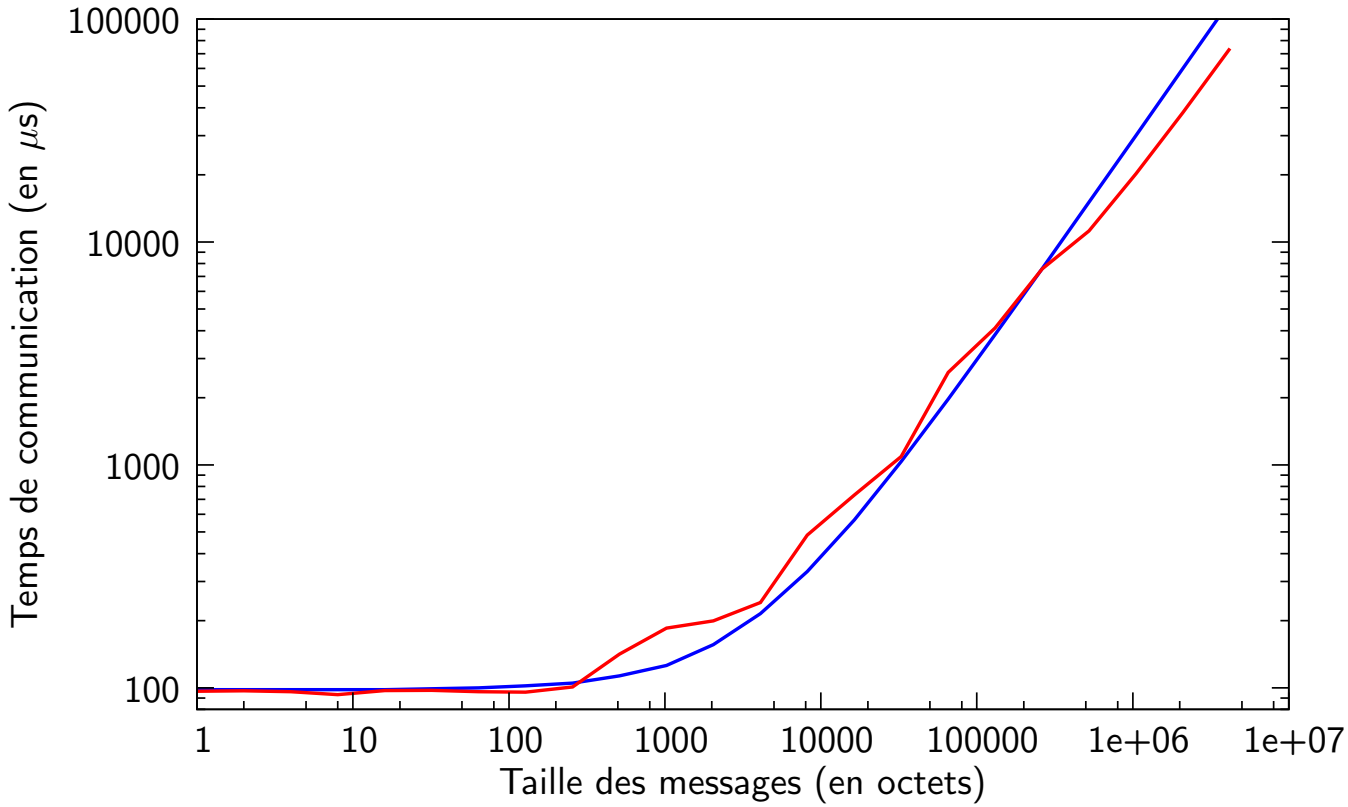
# Communications point-à-point : *Griffon*




Mesures expérimentales avec SKaMPI

Modèle linéaire par défaut, 32.1% d'erreur

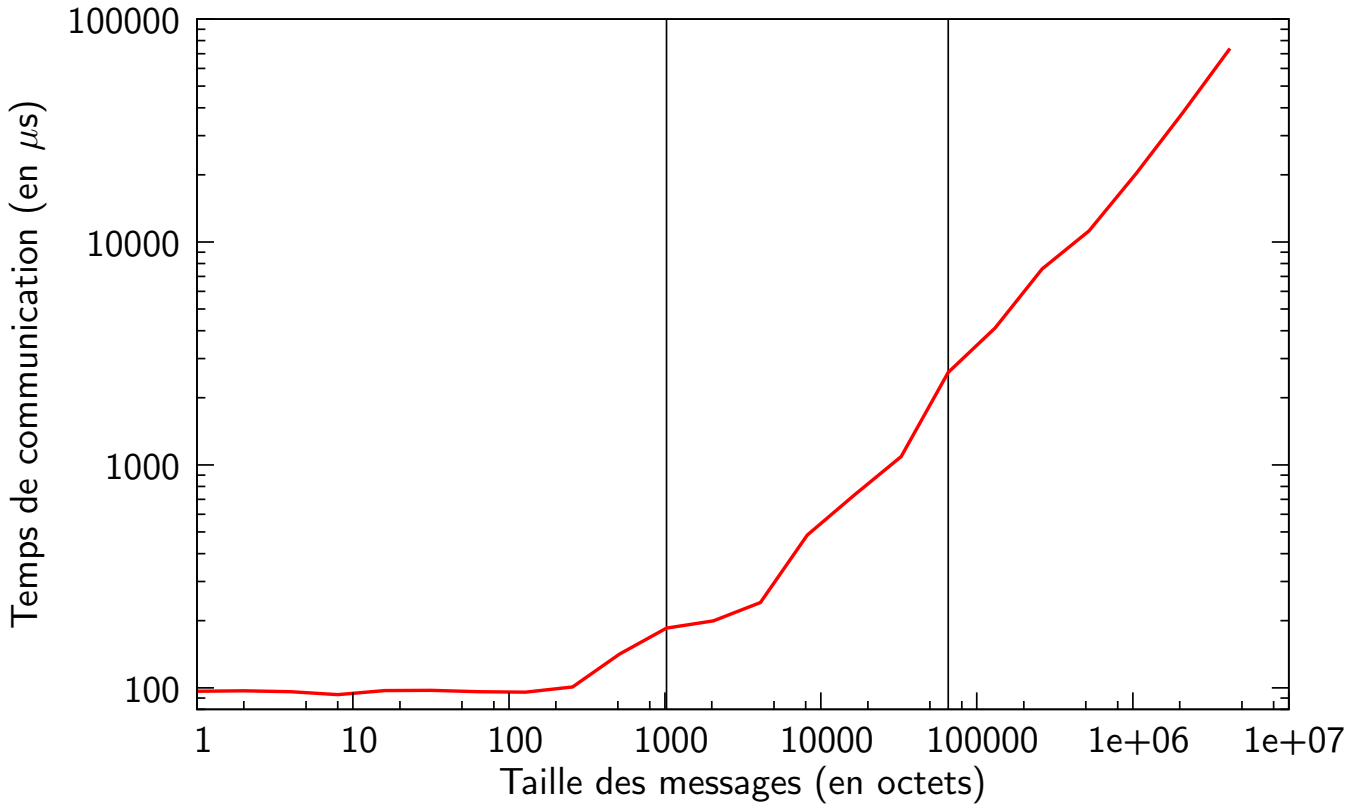
# Communications point-à-point : *Griffon*




Mesures expérimentales avec SKaMPI

Modèle linéaire optimal, 18.5% d'erreur

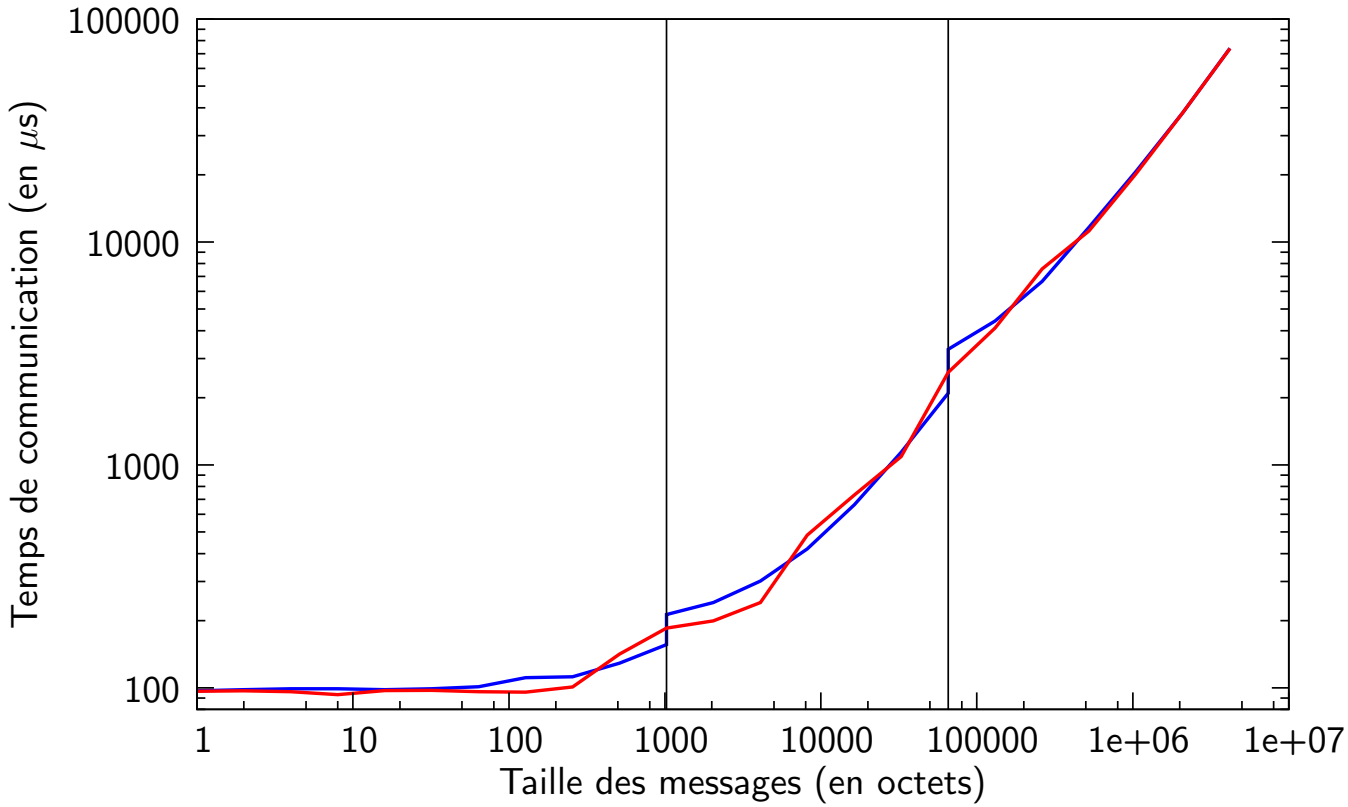
# Communications point-à-point : *Griffon*




Mesures expérimentales avec SKaMPI

Découpage en morceaux

# Communications point-à-point : *Griffon*




Mesures expérimentales avec SKaMPI

Nouveau modèle linéaire par morceaux, 8.63% d'erreur

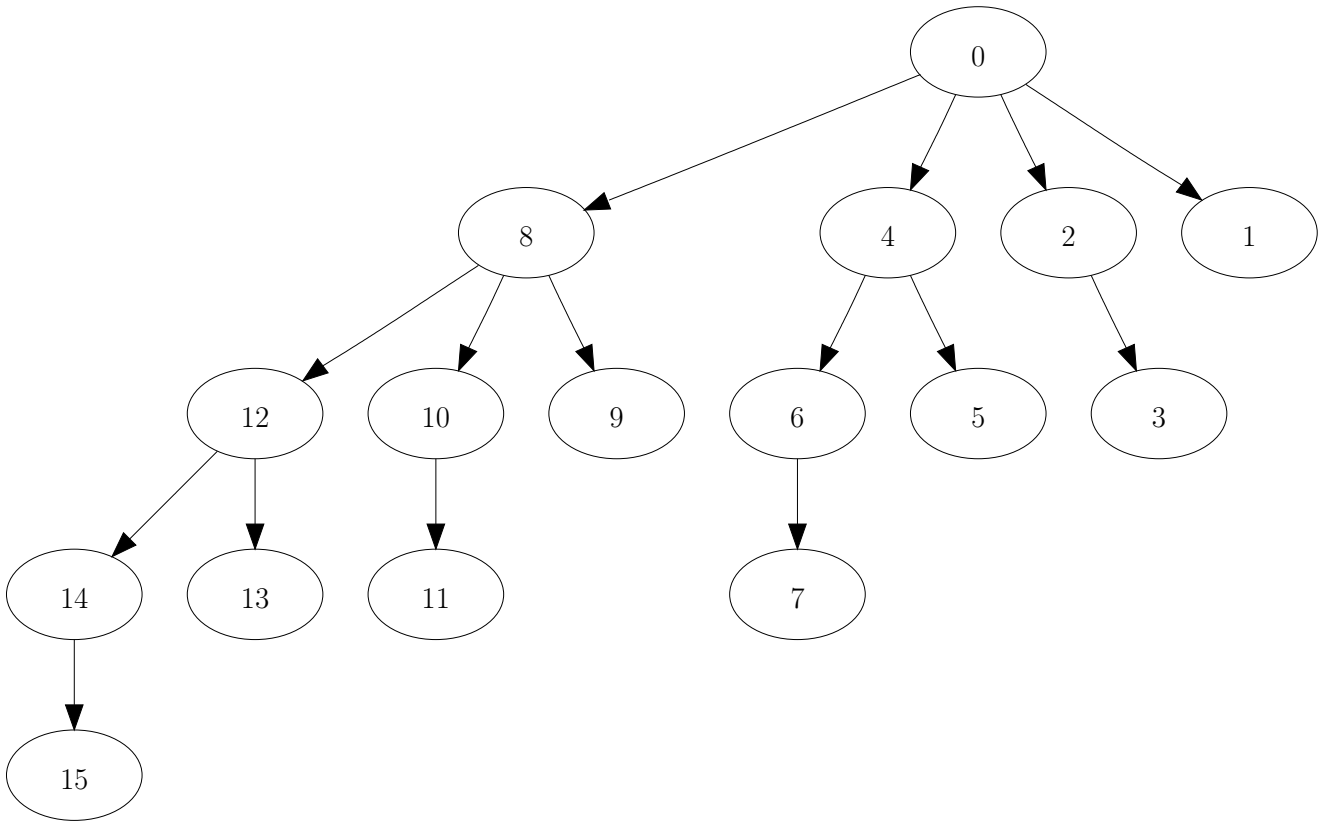
# Instantiation du modèle linéaire par morceaux

- 9 paramètres au lieu de 3 pour le modèle précédent
  - 2 limites de segments
  - 2 facteurs $\alpha$ et $\beta$ par segment
  - 1 facteur $\gamma$ global
- SMPI est fournit avec un script d'instantiation qui utilise
  - 1 fichier de données au format SKaMPI
  - Le nombre de liens traversés lors du ping-pong
  - La valeur de $L$ et $B$ pour les liens
  - Les limites des segments

# Collectives : choix des algorithmes

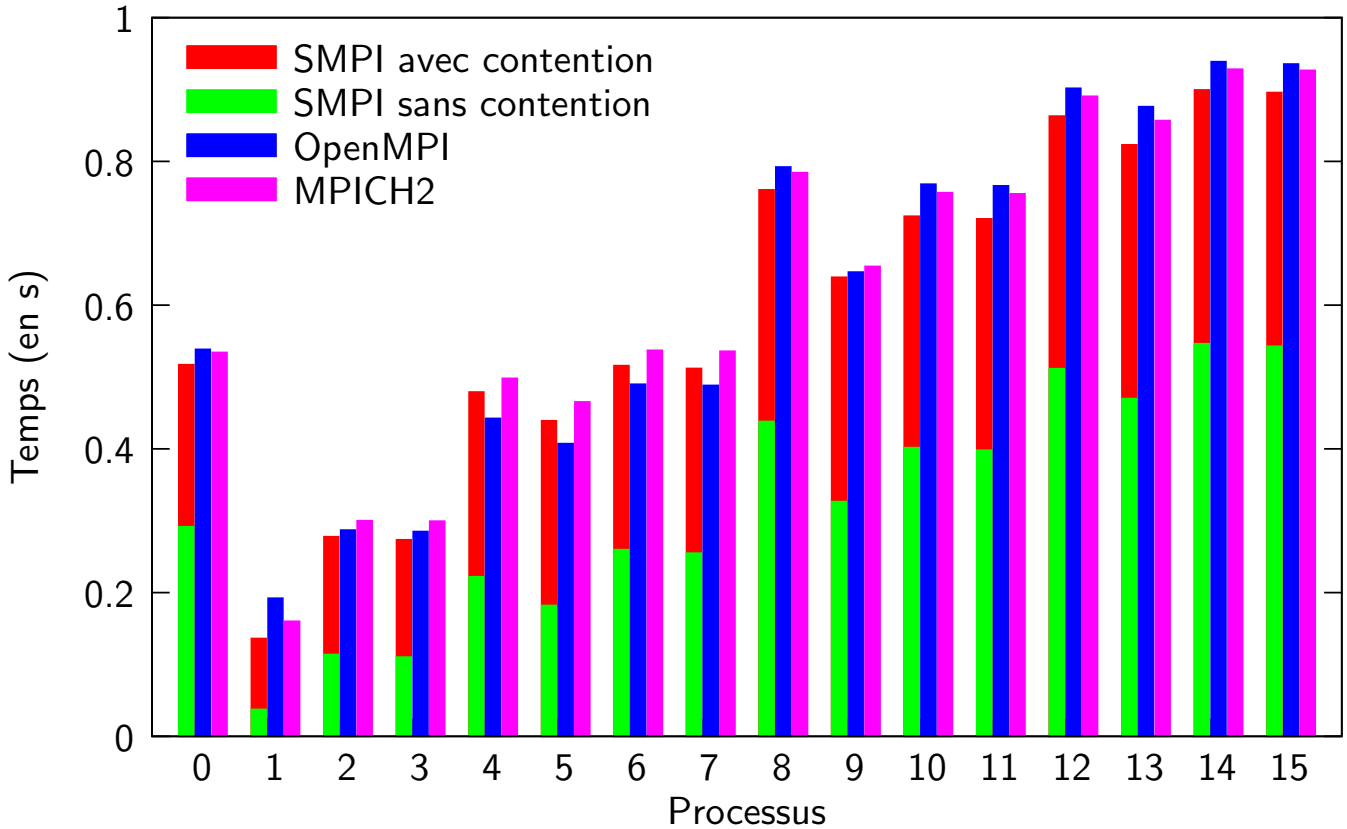
- Différents algorithmes optimaux dans différentes situations
- Monde réel (OpenMPI, MPICH2)
  - Sélection dynamique des algorithmes
  - Fonction du nombre de processus et de la taille des messages
- Monde simulé (SMPI)
  - Un seul algorithme pour tout les cas
- Conséquence : implémentation manuelle pour comparaison

# One-to-many : MPI_Scatter




- Algorithme par arbre binomial
- 64 Mo à la racine, 4 Mo par processus

# Scatter : Comparaison entre processus



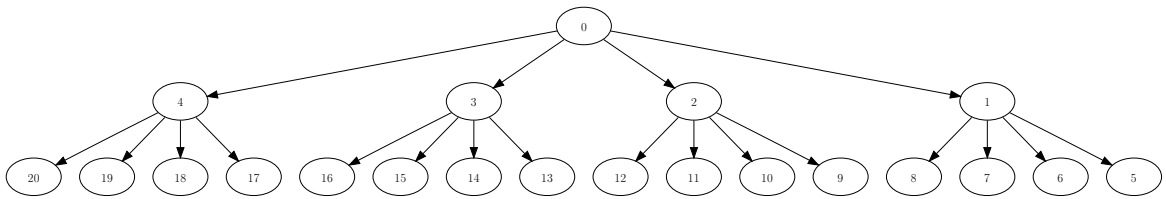

- La prise en compte de la contention est importante
- Comparaison SMPI/MPICH2 $\Leftrightarrow$ OpenMPI/MPICH2 : 5.3% d'erreur

# Plan

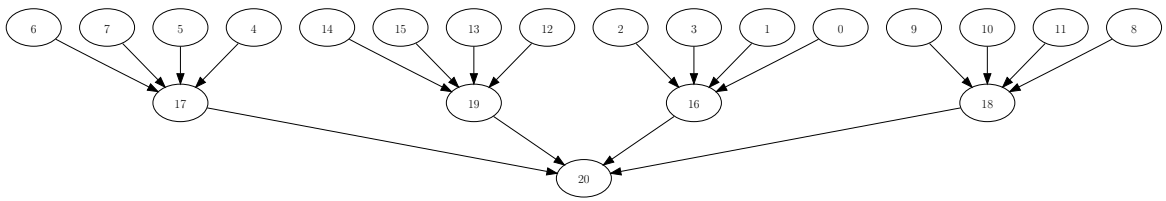
- Introduction
- Précision
- Scalabilité
  - NAS Benchmark Data Traffic
  - Réduction de la consommation mémoire
  - Résultats sur Data Traffic
- Vitesse
- Conclusion et perspectives

# NAS : Data Traffic (DT)

- Peu de calculs (hormis la construction du graphe)
- Schémas de communications possibles :
  - WhiteHole (WH)



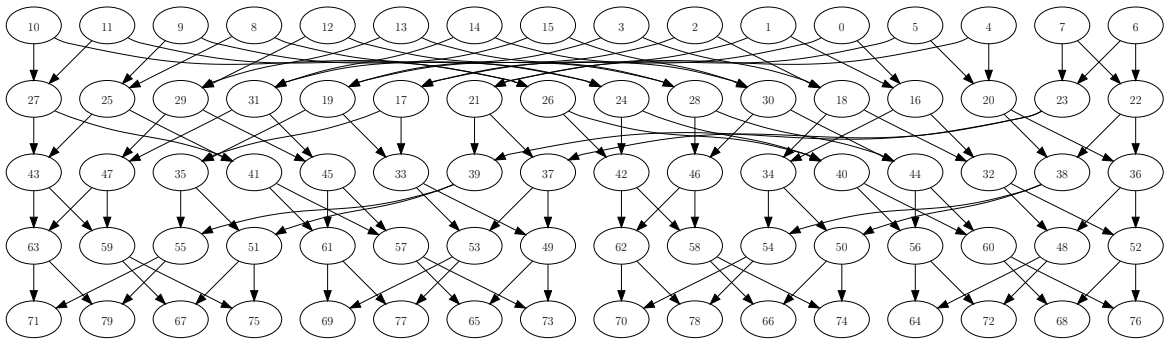

  - BlackHole (BH)




- La taille du problème détermine le nombre de processus
  - Classe A : 21 processus
  - Classe B : 43 processus
  - Classe C : 85 processus

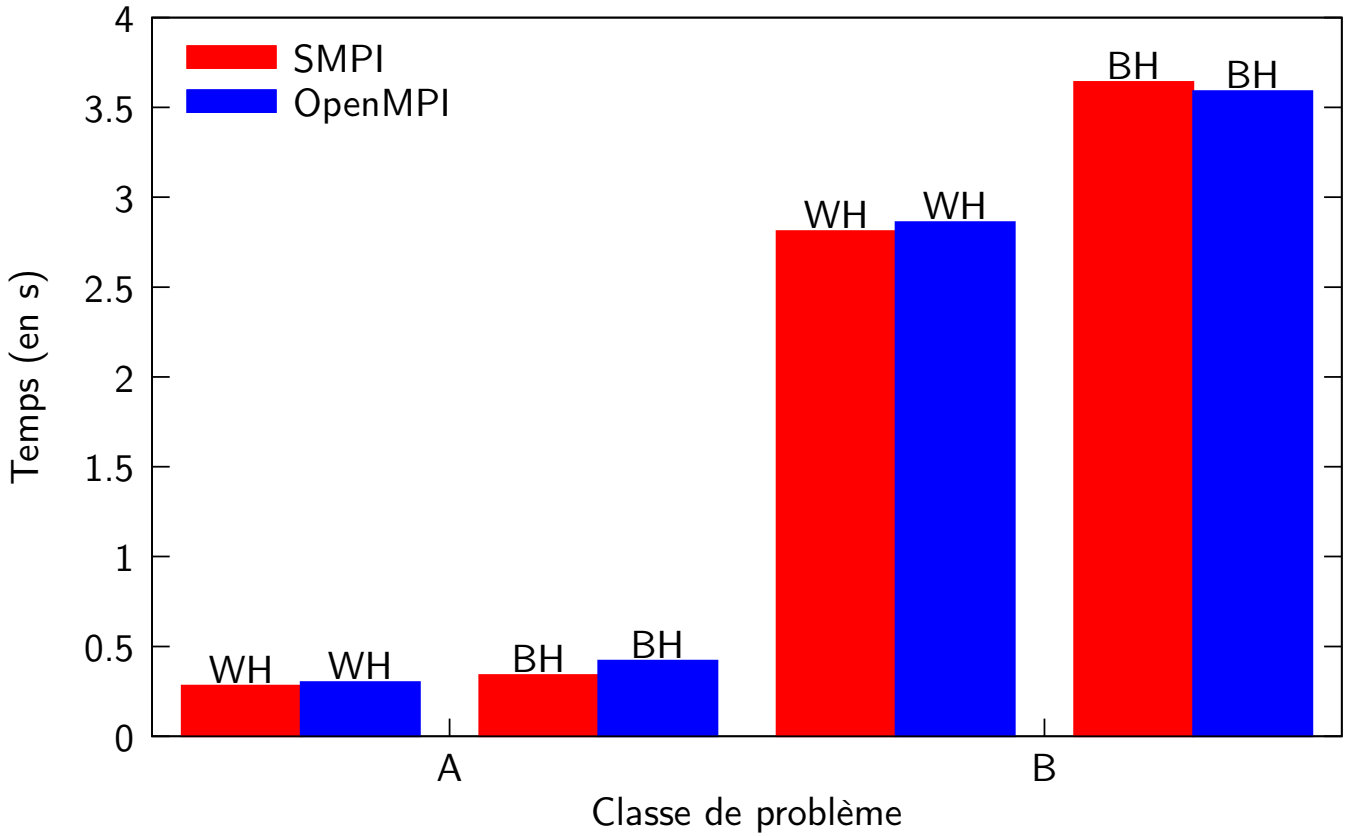
# NAS : Data Traffic (DT)

- Peu de calculs (hormis la construction du graphe)
- Schémas de communications possibles :
  - Shuffle (SH)




- La taille du problème détermine le nombre de processus
  - Classe A : 80 processus
  - Classe B : 192 processus
  - Classe C : 448 processus

# DT : Comparaison avec OpenMPI




- Erreur moyenne : 8.11%
- Difficile d'exécuter SH ou la classe C en réel : taille limite du cluster
- Pas assez de mémoire pour simuler SH au-delà de la classe B

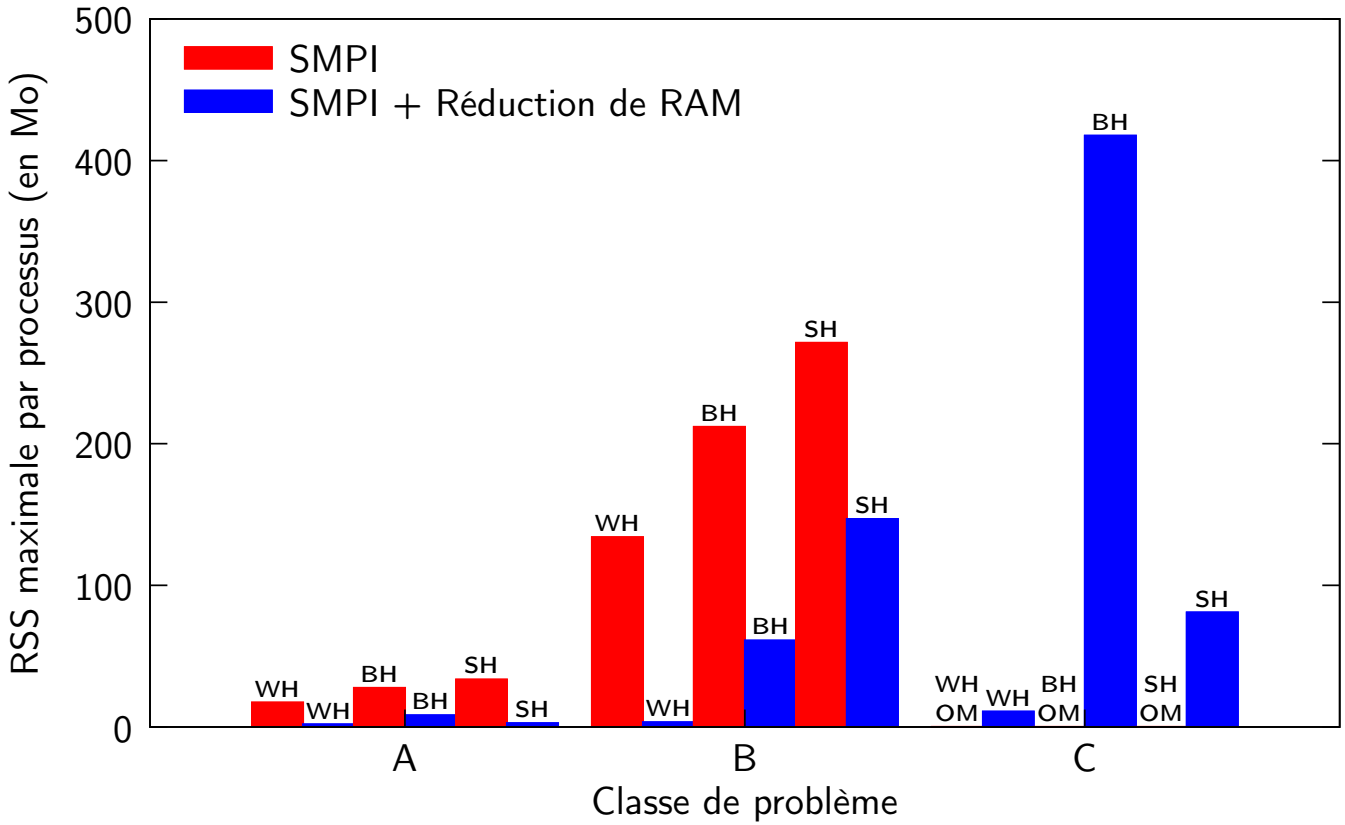
# Réduction de la consommation mémoire

- Exemple :

```
double* data = (double*)SMPI_SHARED_MALLOC(...);
...
SMPI_SHARED_FREE(data);
```

- Changements mineurs
- Tableaux partagés entre les processus
  - Résultats des calculs faux
  - Temps simulés corrects

# Réduction de la consommation mémoire : DT




- Gain en moyenne d'un facteur 11.9 (maximum 40.5)
- La classe C devient accessible

# Plan

- Introduction
- Précision
- Scalabilité
- Vitesse
  - NAS Benchmark Embarassingly Parallel
  - Réduction de l'utilisation CPU
  - Résultats sur Embarassingly Parallel
- Conclusion et perspectives

# NAS : Embarassingly Parallel (EP)

- Pas de communications (hors aggrégation des résultats)
- Partage du calcul entre les processus

- Cas réel idéal : exécution parallèle des processus
- Simulation : exécution séquentielle des processus
- Simulation plus lente que l'exécution réelle avec moins de ressources

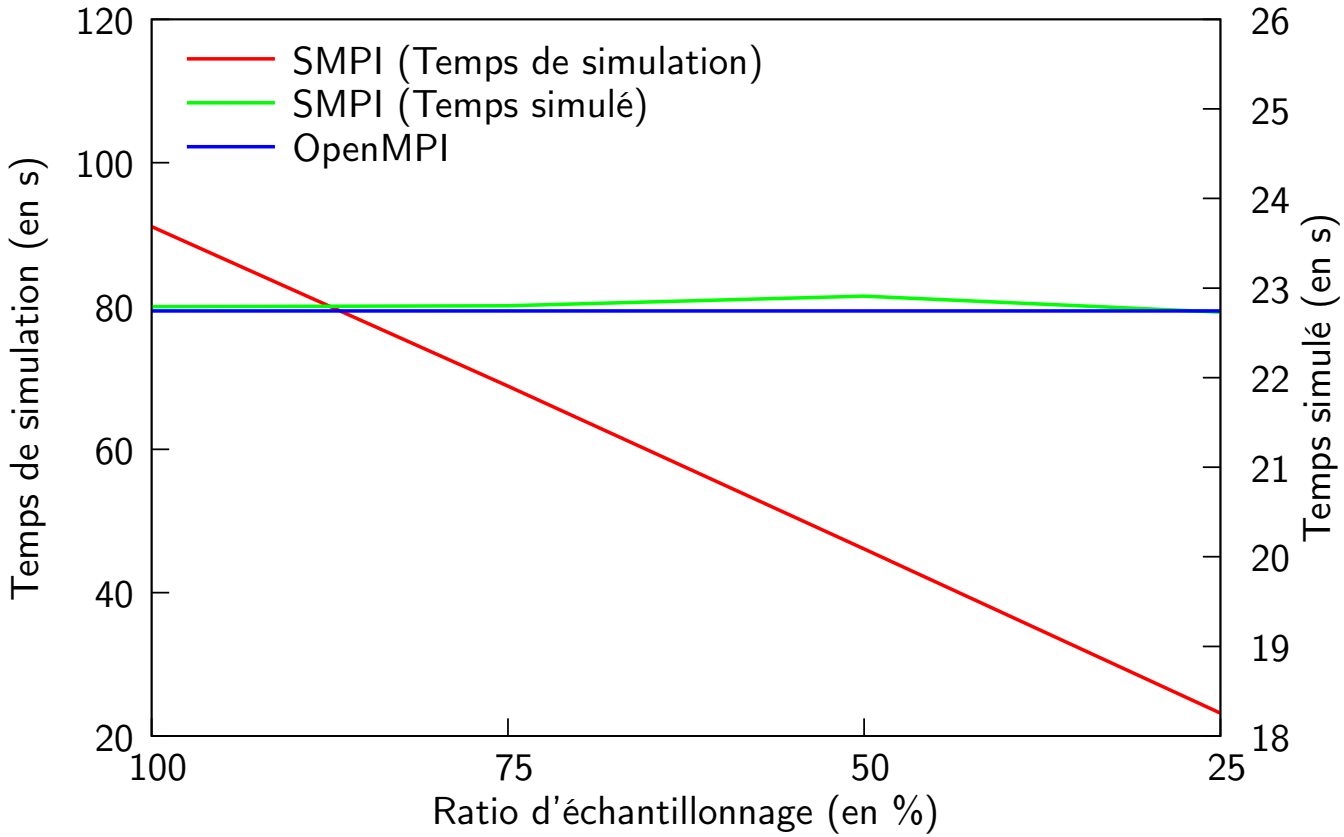
# Réduction de l'utilisation CPU

- Exemple :

```
for(i = 0; i < n; i++) SMPI_SAMPLE_LOCAL(0.75*n,0.01) {
    ...
}
...
for(j = 0; j < k; j++) SMPI_SAMPLE_GLOBAL(0.5*k,0.01) {
    ...
}
```

- Changements mineurs
- Échantillonnage : mesure du temps CPU pour un ensemble d'itérations
  - Chaque processus exécute le nombre d'itérations spécifié (LOCAL)
  - Tous les processus participent à l'exécution (GLOBAL)
- Simulation des itérations restantes par la moyenne des mesures

# Réduction de l'utilisation CPU : EP




- Diminution linéaire de l'utilisation CPU avec le ratio d'échantillonnage
- Pas de perte de précision de simulation

# Plan

- Introduction
- Précision
- Scalabilité
- Vitesse
- Conclusion et perspectives

# Conclusion

- SMPI est un simulateur fonctionnel qui permet la reproductibilité
  - Open Source
  - Script d'instantiation
- Le nouveau modèle réseau apporte plus de précision
- Les techniques de réduction de l'usage des ressources améliorent à la fois la vitesse et la scalabilité

- Les améliorations apportées ont fait l'objet d'une validation importante
  - Connaissance des cas positifs
  - Identification des points améliorables

# Perspectives

- Améliorations à court terme
  - Prise en compte de comportements ethernet
  - Prise en compte de phénomène de sérialisation des paquets
- Objectifs à moyen terme
  - Modèles pour d'autre types de réseaux : Myrinet, Infiniband
- Objectifs à plus long terme
  - Simuler les entrées/sorties
  - Utiliser automatiquement les techniques de réduction à la compilation
  - Simuler une implémentation MPI donnée : OpenMPI, MPICH2