Portage de StarPU sur la bibliothèque de communication NewMadeleine

Guillaume Beauchamp

Université de Bordeaux
Encadré par Alexandre Denis,
en collaboration avec Nathalie Furmento, Olivier Aumage, Samuel Thibault,Emmanuel Agullo
dans l'équipe Inria Tadaam.

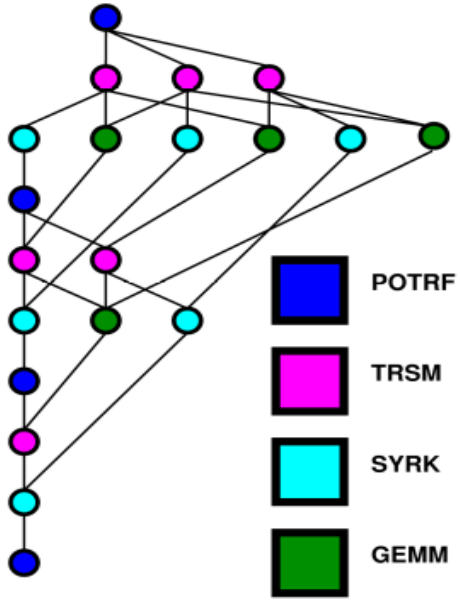
13 septembre 2017

Introduction

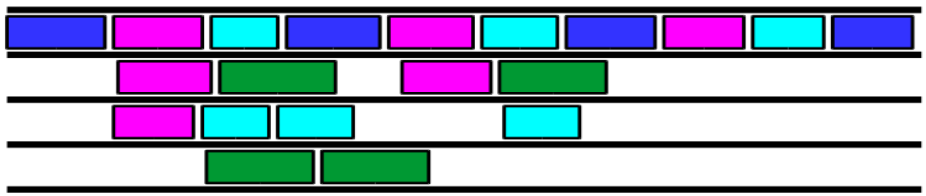
- De nombreux domaines(aéronautique, météorologie,...), on besoin d'effectuer des calculs longs et complexes dans des temps raisonnables.
- Pour la plupart de ces applications, la précision de la simulation est liée à la quantité de calcul effectuée.
- Répartir le calcul à l'intérieur d'une machine et entre plusieurs machines.
- Architecture hétérogène : Cœurs et Accélérateurs (GPU, Xeon Phi)

Plan

StarPU



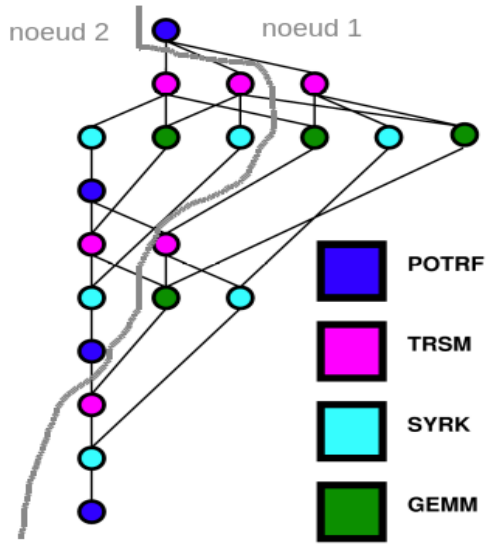
- Ordonnanceur de tâches multisupport(cœurs, accélérateurs)
- Effectue les communications intra-nœud nécessaires
- Graphe de tache maintenant la cohérence de données



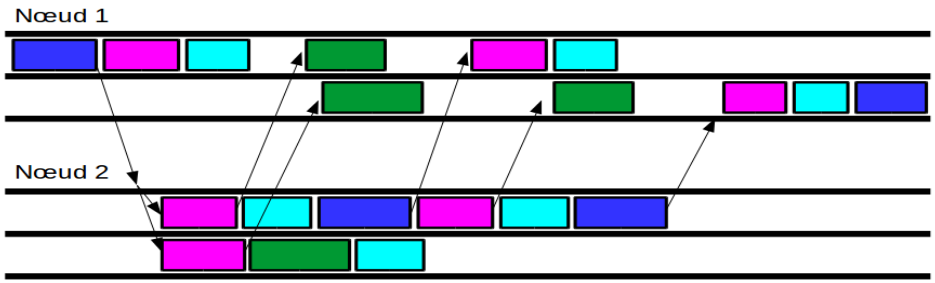
MPI

- Communication inter-nœuds.
- Une interface ayant de multiples implémentations (OpenMPI, IntelMPI,...)
- Modèle send receive (communicateur, tag, destination)
- Communications non bloquantes souvent utilisées pour recouvrir les communications.

StarPU-MPI



- Back-end de StarPU permettant d'effectuer des communications inter-nœuds
- Avec tâche : communications implicites. Chaque nœud a un graphe de tache.
- Utilisation possible sans tâches de manière similaire a MPI.

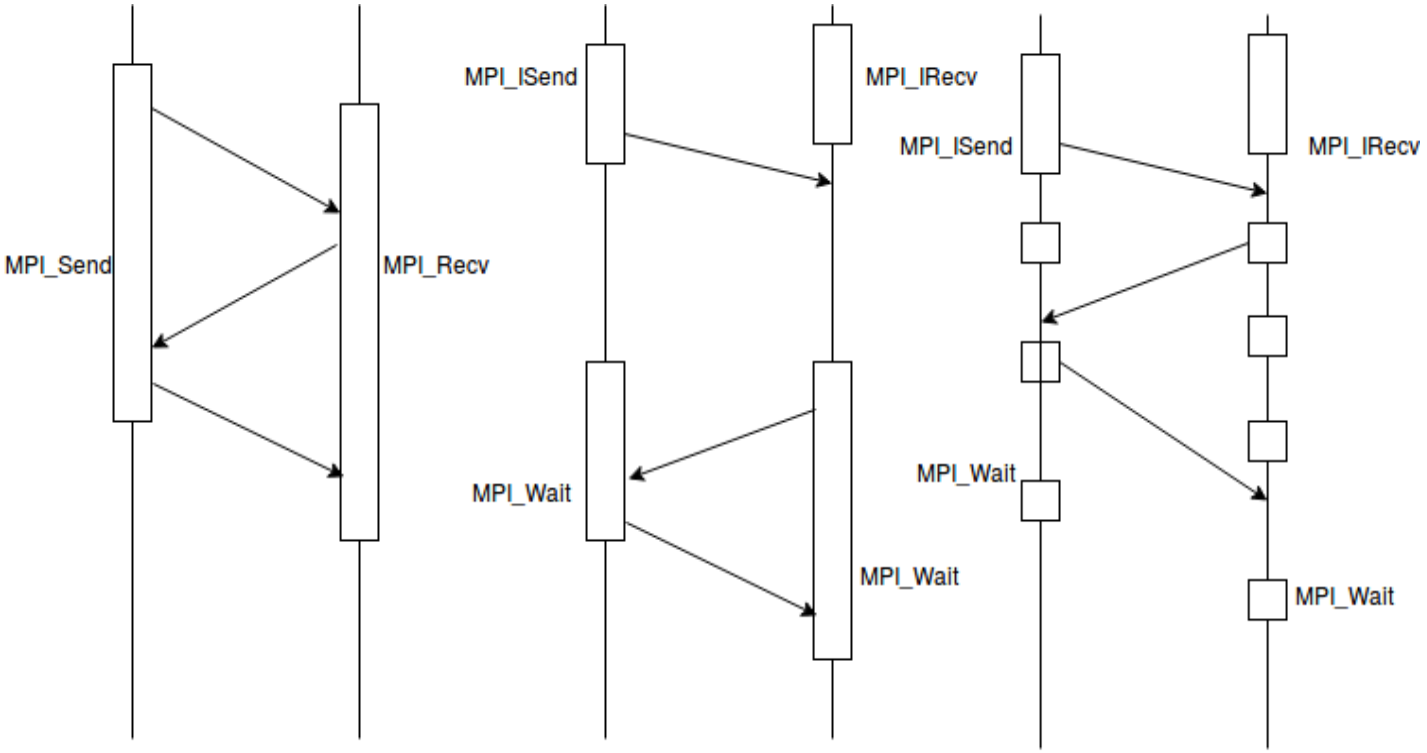


New Madeleine

- Communication inter et intra-nœuds. Fournit une implémentation MadMPI de l'interface MPI.
- Événementiel : RPC ou Send Receive (avec possible callback)
- Peut fusionner des communications ayant la même destination.
- Support appels concurrents à la librairie.
- Progression garantie par NewMadeleine(idle, timer, polling)

Problématique

- Manque de scalabilité en nombre de requêtes des implémentation MPI
- Les communications non bloquantes de MPI ne garantissent pas la progression.
- Thread de communication et impact sur le recouvrement



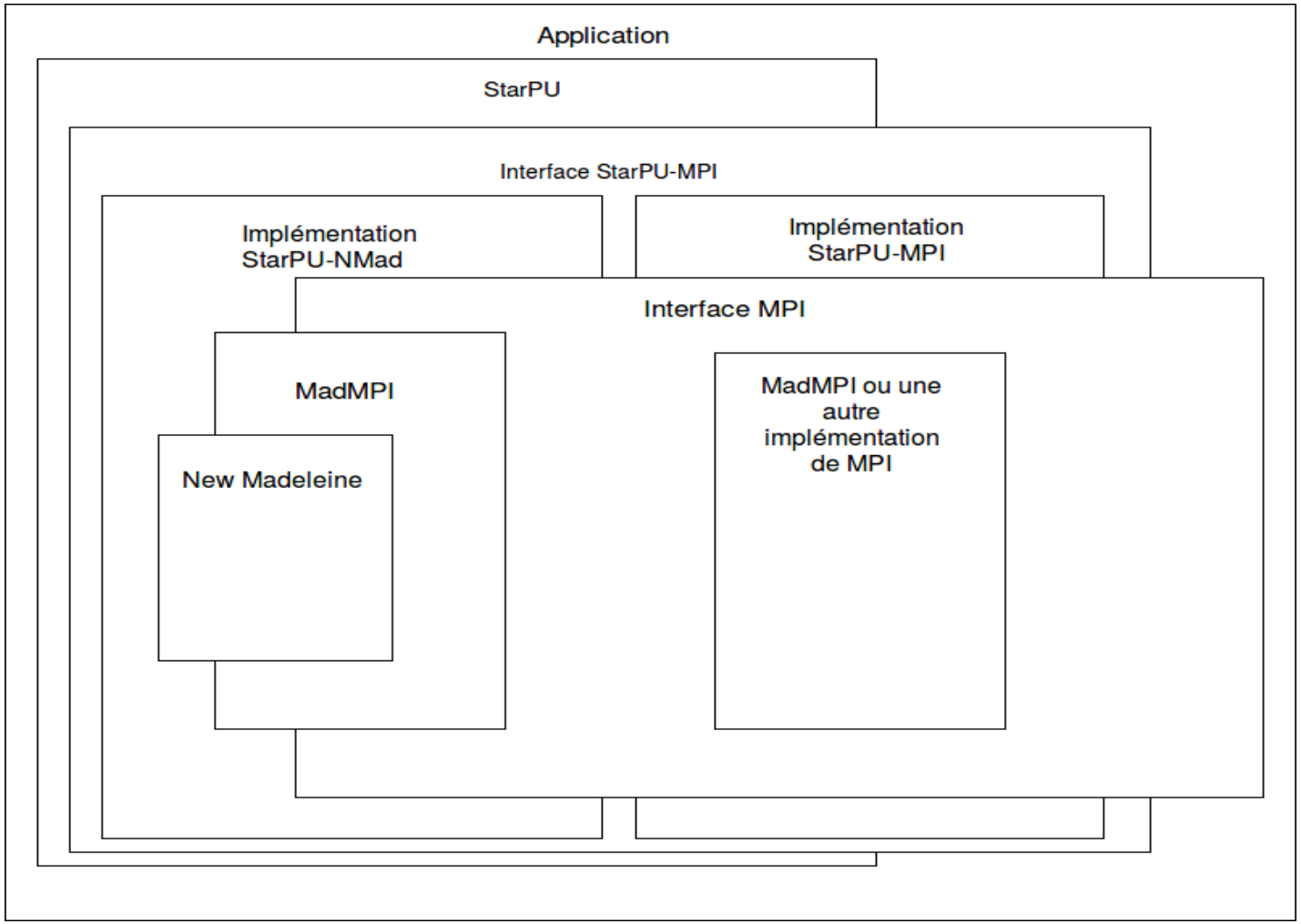
Communication bloquante

Communication non bloquante

Communication non bloquante, avec appels periodiques de la librairie MPI

Travail effectué

- Portage de StarPU-MPI sur MadMPI puis NewMadeleine
- Passage en événementiel : plus de thread de progression
- Étude des performances de StarPU-NMad.



Modifications de Starpu-MPI à StarPU-NewMadeleine

- MadMPI convertit les types de MPI à NewMadeleine.
- Utilisation événementielle de NewMadeleine : send receive + callback NewMadelaine
- Pas besoin d'un thread de communication
- Thread de callback utilisateur.

Send/Recv de StarPU-MPI

StarPU-MPI send/receive

Pousser paramètres dans la pile des nouvelles requêtes

Algorithm 1: isend/recv de StarPU-MPI

```
foreach req dans nouvelles_requêtes do
    MPI_ISend/Recv (req)
    Retirer req de nouvelles_requettes
    Ajouter req dans requêtes_en_cours
end
foreach req dans requêtes_en_cours do
    MPI_Test (req) if req terminée then
        if callback utilisateur demandé then
            exécuter callback utilisateur
        end
        marquer données comme reçues
    end
end
```

Algorithm 2: Thread de communication de StarPU-MPI

StarPU-NewMadeleine send/receive :

MadMPI empaquette les données selon le format décrit par MPI

MadMPI convertit le communicateur, le tag, et le destinataire MPI en leur équivalant NewMadeleine

Début isend/recv NewMadeleine

Ajout callback de terminaison de requête

Algorithm 3: isend/recv de StarPU-NMad

```
if callback definit par l'utilisateur then
    Empiler callback
    Réveiller thread de callback
else
    Marquer les données comme reçues et la requête
      StarPU-MPI comme terminée
end
```

Algorithm 4: Callback de terminaison appelé par NewMadeleine

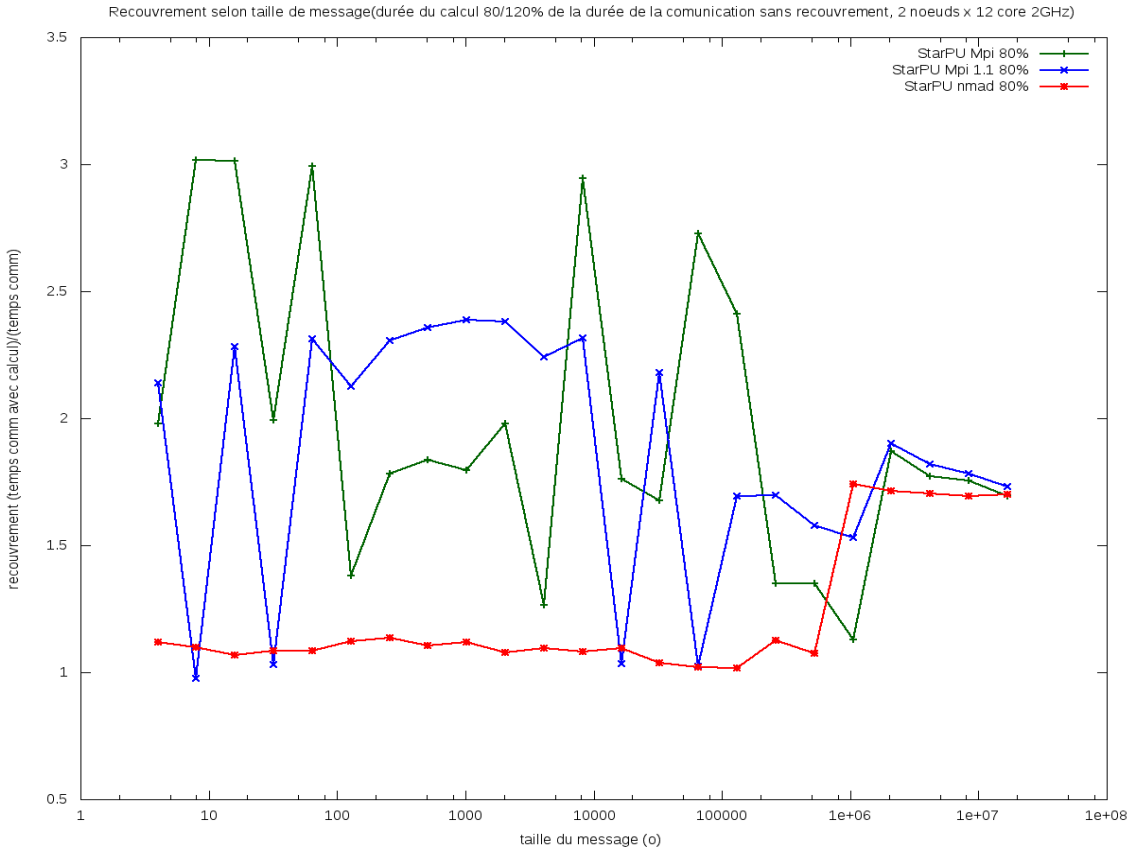
```
while starpu vivant ou requêtes actives do
    Attendre callback
    Dépiler callback
    Exécuter callback
    Marquer données comme reçues
    Libérer requête
end
```

Algorithm 5: Thread de callback

Performances

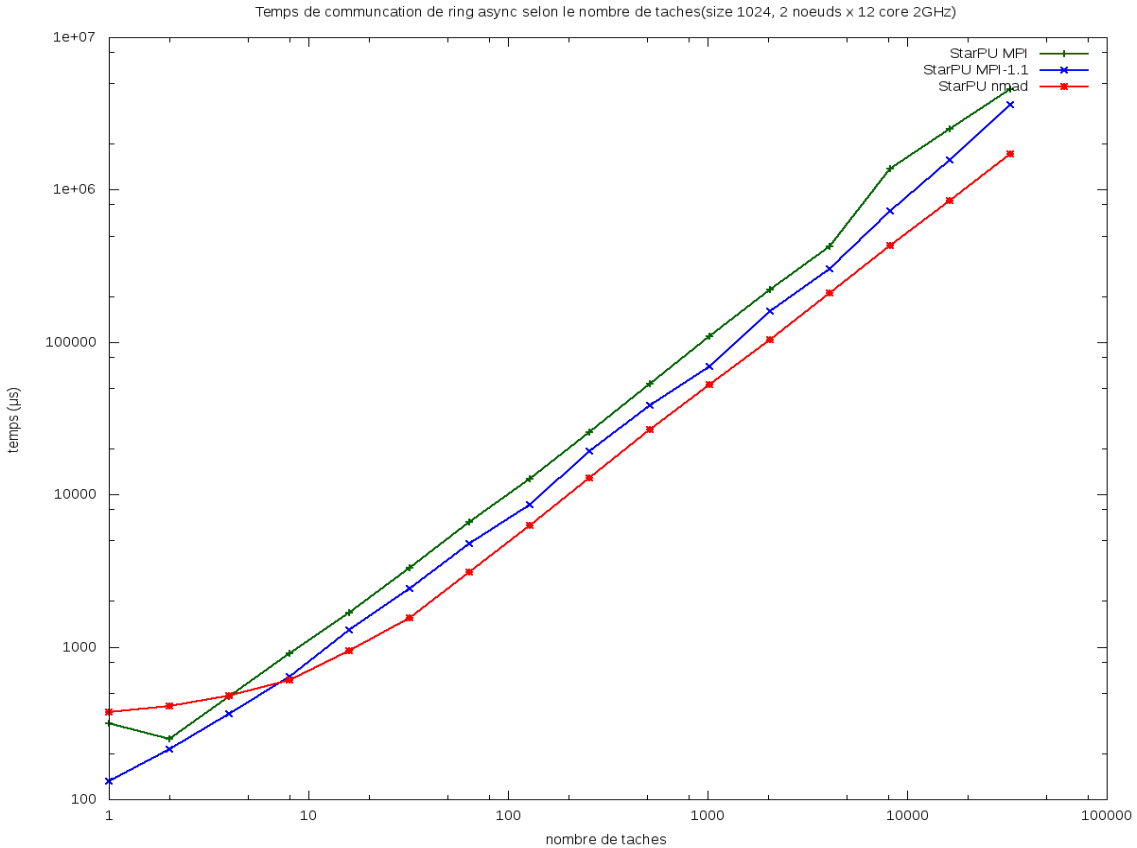
- Deux nœuds intel de 12 coeurs 2GHz
- Relié par un réseau Infiniband
- Nous comparons nos performances a celles de StarPU 1.1 et 1.3 avec IntelMPI pour implémentation de MPI.

Performances : Recouvrement



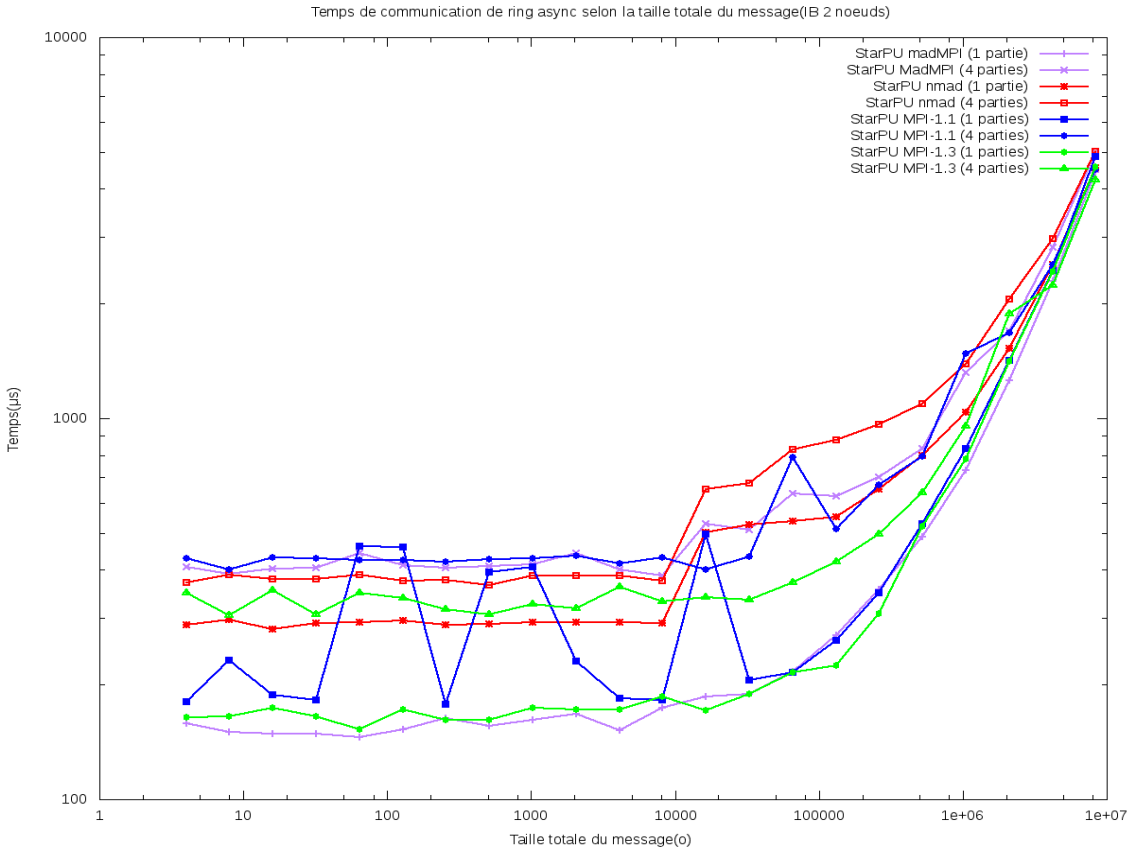
- Recouvrement : ratio temps calcul et communication/temps communication (dans l'idéal vaut 1).
- StarPU-MPI a un recouvrement équivalent à une implémentation séquentielle.
- StarPU-NMad sauf pour des messages de très grande taille, a un très bon recouvrement (et stable)

Performances : Scalabilité en nombre de requêtes



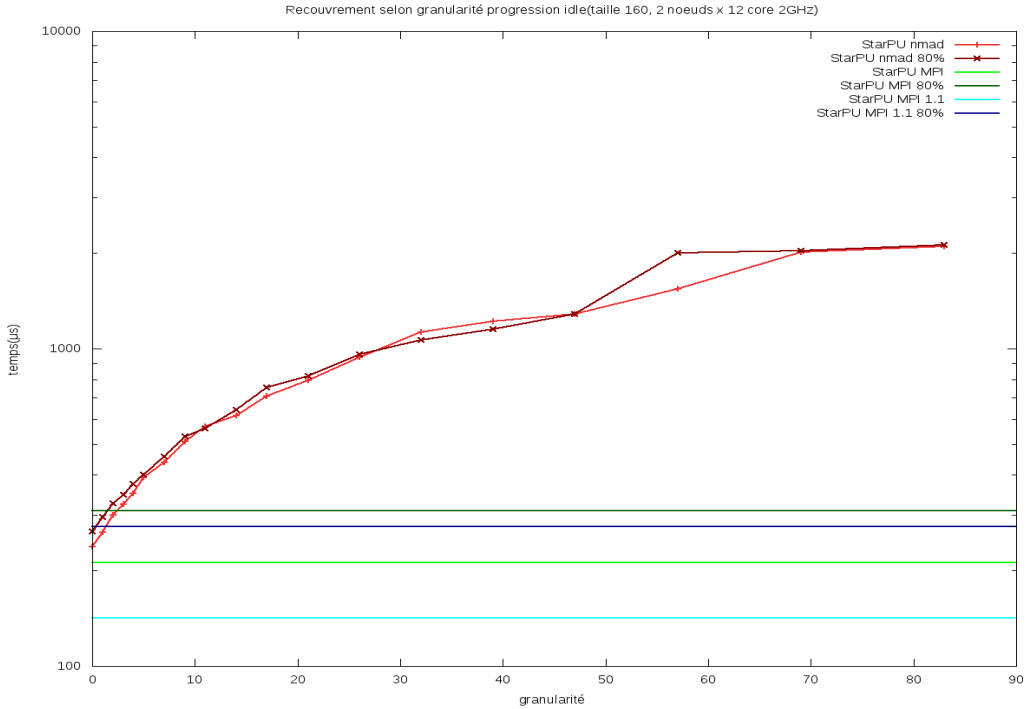
- Intervalle entre deux communications jusqu'à 3 fois plus faible pour un grand nombre de requêtes avec l'implémentation utilisant NewMadeleine.

Performances : Latence



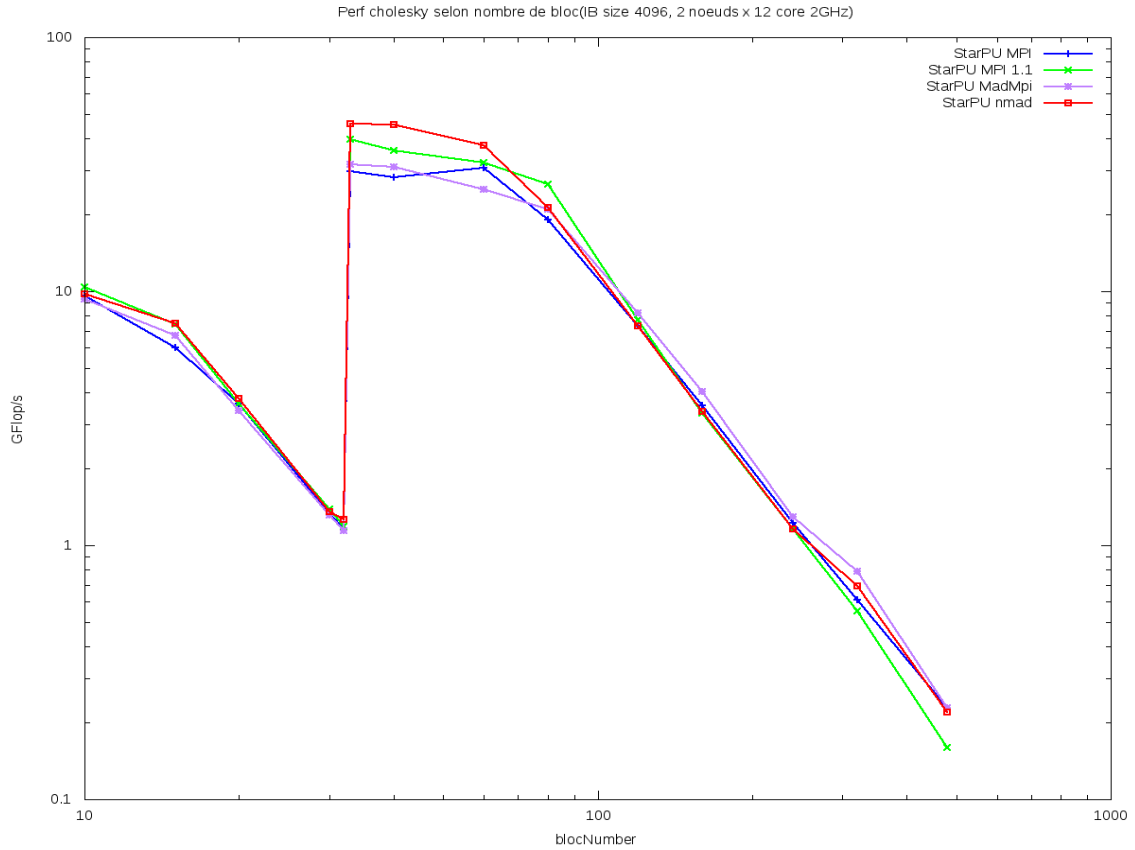
- L'implémentation NMad a une latence bien plus élevée pour l'envoi d'un seul message
- Sa fusion des message réduit fortement le surcoût d'un envoi en plusieurs parties.

Performances : Impact de la granularité



- Le thread idle de NewMadeleine effectue la majorité de la progression.
- Granularité (durée minimale entre 2 progression) par défaut du thread idle de NewMadeleine : 5µs.
- Latence de NewMadeleine proportionnelle à la granularité, mais largement supérieure a celle demandée.
- Meilleure latence de StarPU-NMad équivalente à celle de StarPU-MPI.
- Une granularité réduite (<5) augmente légèrement le surcoût du recouvrement qui reste bien meilleur que celui de l'implémentation MPI.

Performances : Factorisation de matrice Cholesky



- Pour la taille de bloc optimale, NewMadeleine a des performance 25% meilleures que celle des implémentation MPI, grâce à son meilleur recouvrement.
- Pour d'autres tailles de blocs, les performances sont similaires, le temps de calcul étant négligeable par rapport à celui des communications.

Conclusion

- StarPU-NewMadeleine a une meilleure scalabilité en nombre de requêtes, et un meilleur recouvrement.
- En réduisant la granularité elle a une latence équivalente a l'implémentation MPI, avec une faible dégradation de ses autres performances.

Travaux futurs :

- Réduire surcoût en latence de la progression idle de NewMadeleine.
- Étudier l'impact de notre implémentation sur la performance d'applications réelles comme Chaméleon.

Merci de votre attention

Questions