TER

Simulation d'application dynamiques pour plateformes de calculs hautes performances

Équipe MESCAL

Steven QUINITO MASNADA

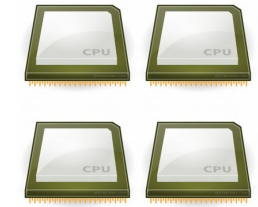
Grenoble, 8 Juin 2015

Topic

1. Contexte et problématique
2. Évaluation des performances en HPC
3. Analyse du problème
4. Méthodologie
5. Contribution
6. Validation
7. Conclusion

Architecture et standards HPC

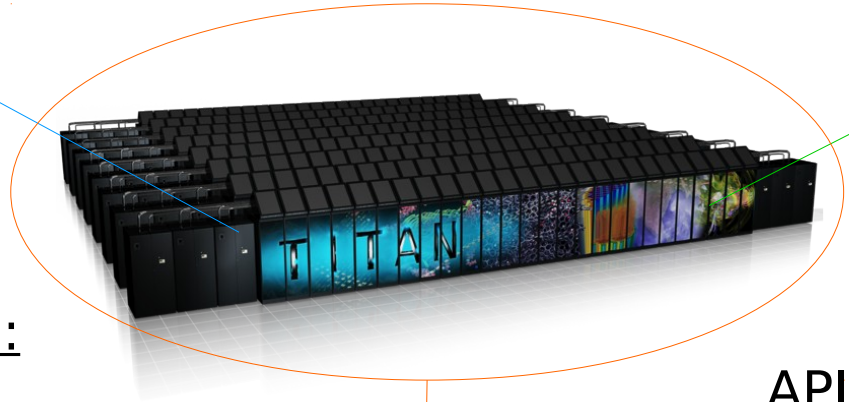
Multicœurs



OpenMP

API multithread :

- De plus haut niveau que PThread,
- Permet d'exploiter les architectures multicœurs
- Facilite le découpage des traitements



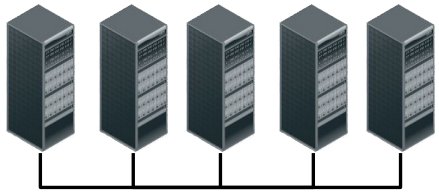
Hybride

NVIDIA CUDA

API de caculs sur GPU :

- Spécifier calculs GPU
- Synchroniser CPUs - GPUs

Clusters



MPI

API de communication :

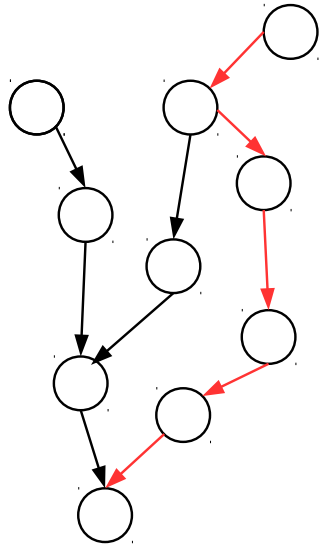
- De plus haut niveau que les sockets,
- Mécanismes de comminucation supplémentaires (exemple broadcast)

Limites des approches classiques

- Utiliser plusieurs paradigmes $\rightsquigarrow$ programmation complexe
- Exemple pour exploiter efficacement un GPU sur un seul noeud:
  - transférer données du CPU au GPU,
  - lancer le calcul sur le GPU
  - gérer synchronisation pendant attente résultat
  - occuper CPU
  - récupérer résultat.
- Et avec plusieurs noeuds?
- Statique, système réglé comme un horloge $\rightsquigarrow$ pas portable.
- Solution: Dynamique mais presque impossible avec APIs classiques.

Nouvelle approche: Paradigme de tâches

- Nouvelle abstraction: les tâches
  - Plus besoin de se soucier de la ressource sur laquelle le traitement est effectué.
  - Exprimer calcul en graphe de tâches $\rightsquigarrow$ système dynamique plus simple.



- Librairie StarPU:
  - Système runtime
  - basé sur le paradigme de tâches $\rightsquigarrow$ graphe de dépendances.
  - Ordonnancemment dynamique et opportuniste.
- Problèmatique : Performances difficiles à évaluer
  - Configuration runtinme, heuristique, politique ordonnancement.
  - Configuration application, découpage des tâches.

Topic

1. Contexte et problématique
2. Évaluation des performances en HPC
3. Analyse du problème
4. Méthodologie
5. Contribution
6. Validation
7. Conclusion

Test sur systèmes réels

- Exécution réelle sur la plateforme cible $\rightsquigarrow$ coûteux, difficulté d'accès.
- Exéuction non déterministe nécessite de réaliser beaucoup d'expériences $\rightsquigarrow$ extrapolations difficiles.

Simulation

Généralités

- Utilisation de modèles pour prédire comportements.
- Permet s'affranchir de la plateforme $\rightsquigarrow$ peu coûteux.
- Contrôle paramètres $\rightsquigarrow$ systèmes déterministes, extrapolation simplifiée.
- Exécution plus courte.

Simulation par rejeu de trace

Exécution post-mortem: pas adapté ici car flot de contrôle non déterministe.

Hybride simulation / émulation

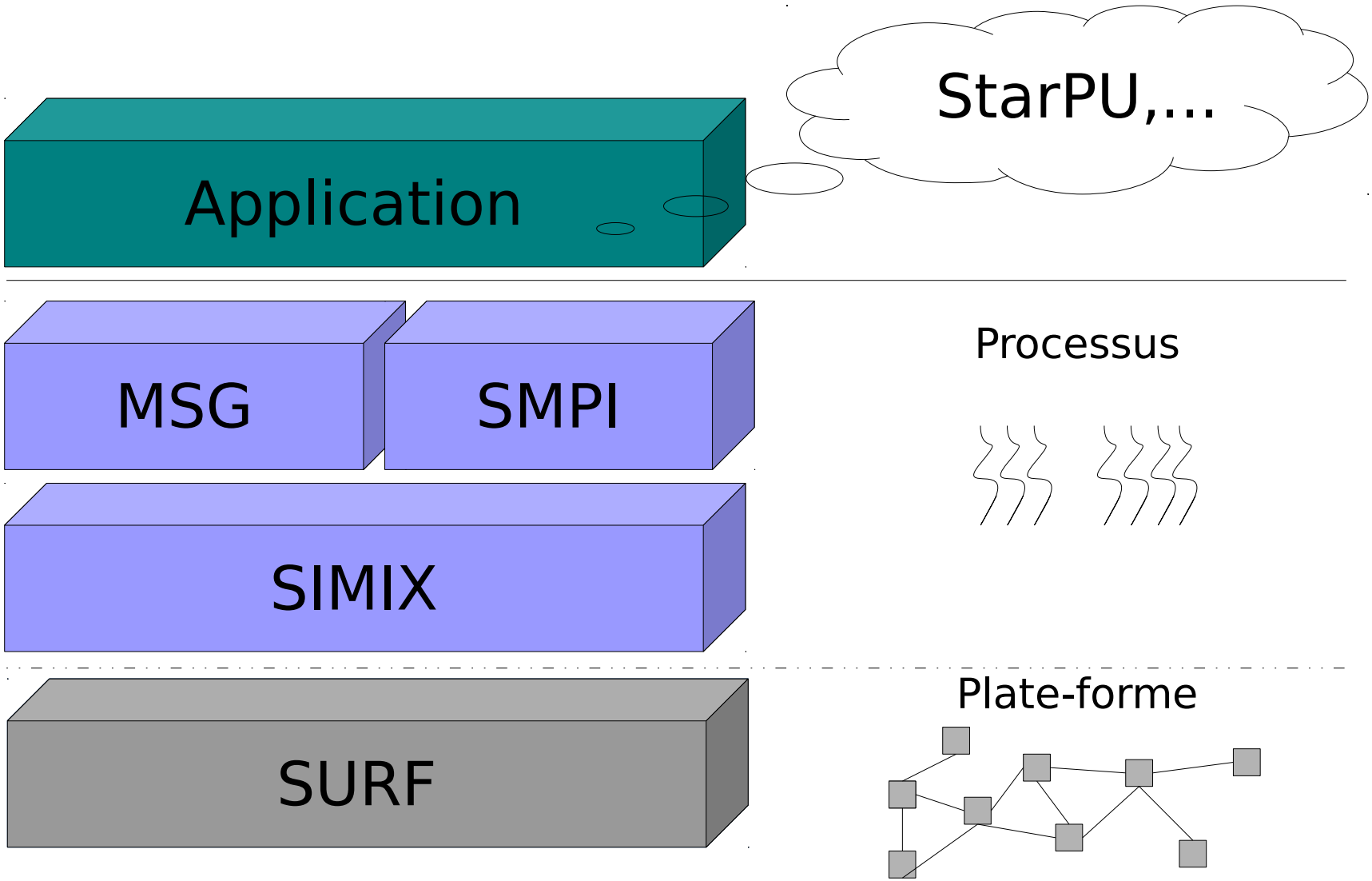
- Simuler plateforme et OS.
- Emuler de l'application.

StarPU porté récemment au dessus de Simgrid.

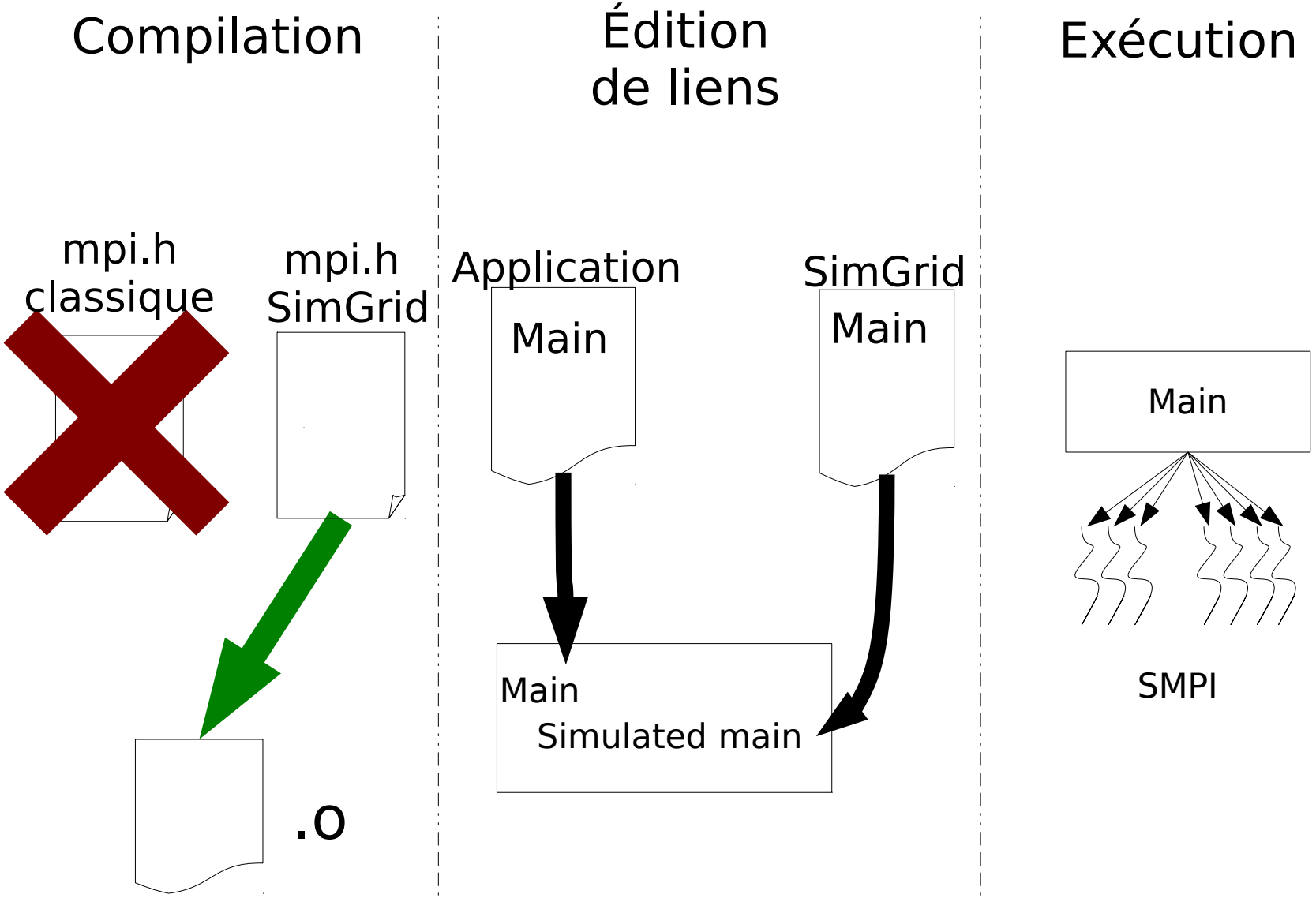
Topic

1. Contexte et problématique
2. Évaluation des performances en HPC
3. Analyse du problème
4. Méthodologie
5. Contribution
6. Validation
7. Conclusion

SimGrid: Généralités

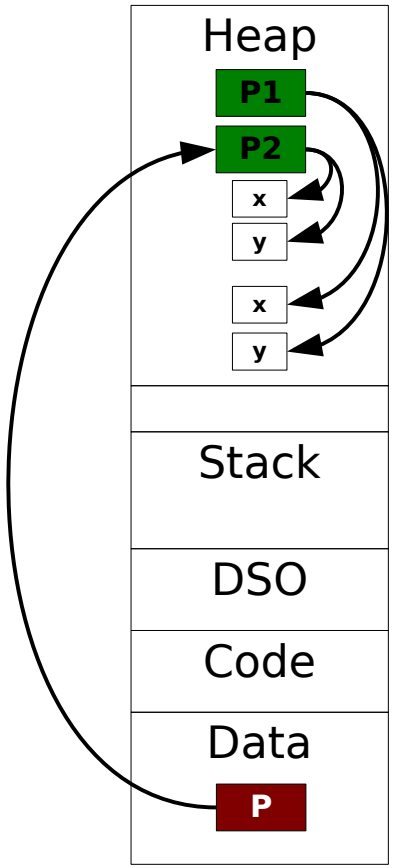


Construction de l'application MPI simulée



Privatisation segment data

- Dans SimGrid les processus sont modélisés par des threads $\rightsquigarrow$ espace adressage partagé.
- MPI environnment à mémoire distribuée.
- Mécanisme de privatisation: ségment virtuel (mmap)



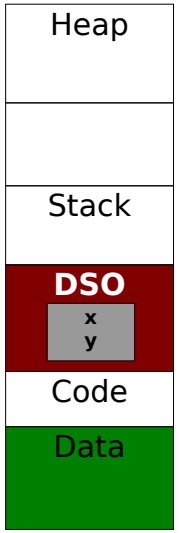
StarPU MSG

- Première version StarPU $\rightsquigarrow$ hybride
- Basé sur MSG car modèle de performance plus proche (communications, environnement mémoire partagé), CPUs GPUs.
- Simulation: calculs, allocations mémoire des tâches, transfert CUDA.

StarPU SMPI: Difficultés de mise en oeuvre

- Besoin de 2 modèles de performances différents à la fois:
  - MSG Intra noeuds $\rightsquigarrow$ mémoire partagée $\rightsquigarrow$ partage.
  - SMPI extra noeuds $\rightsquigarrow$ mémoire distribuée $\rightsquigarrow$ privatisation.
- MSG et SMPI normalement pas utilisés ensemble $\rightsquigarrow$ initialiser correctement les 2.

- Problème des librairies dynamiques.



Topic

1. Contexte et problématique
2. Évaluation des performances en HPC
3. Analyse du problème
4. Méthodologie
5. Contribution
6. Validation
7. Conclusion

Techniques et étude de l'existant

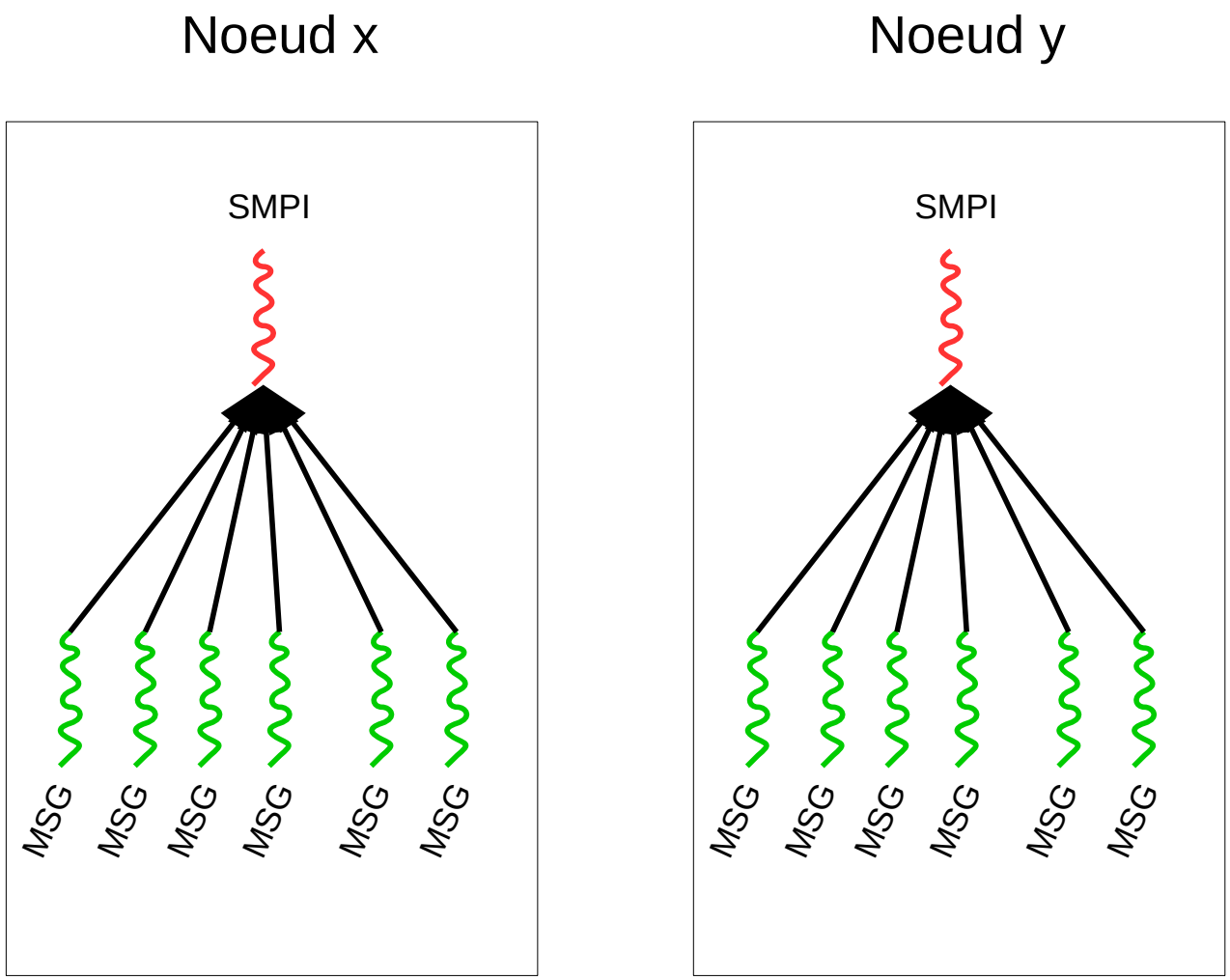
- Dépôt git submobules:
  - StarPU SMPI:
    - SimGrid
    - StarPU
- Suivi:
  - Cahier de laboratoire org mode github.
- Compréhension:
  - Documentation.
  - SimgGrid = 106 350 lignes de codes.
  - StarPU = 172 251 lignes de codes.
  - "Code mining" et vérifications: GDB, Valgrind.

Topic

1. Contexte et problématique
2. Évaluation des performances en HPC
3. Analyse du problème
4. Méthodologie
5. Contribution
6. Validation
7. Conclusion

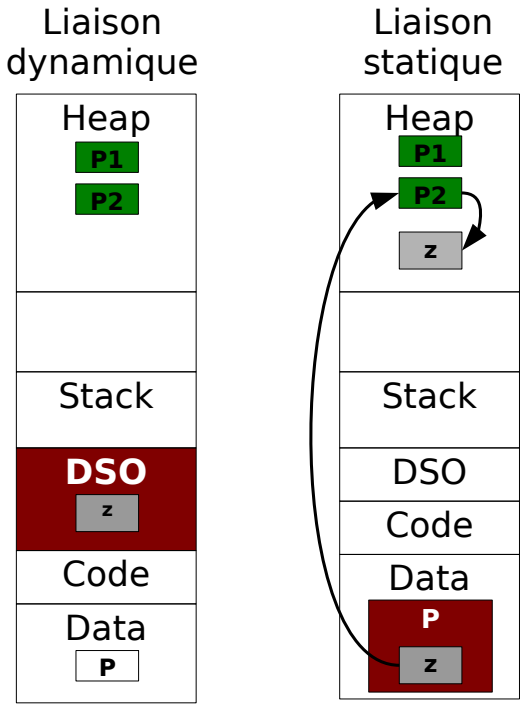
Modification de SimGrid

- Initialisation MSG + SMPI
- Gestion segment data



Librairie dynamiques et modifications StarPU

- Librairies dynamiques:
  - Utilisation librairies statiques.
- Modification StarPU:
  - Initialisation, car privatisation tardive.



Topic

1. Contexte et problématique
2. Évaluation des performances en HPC
3. Analyse du problème
4. Méthodologie
5. Contribution
6. Validation
7. Conclusion

Validation

- Test simple: Modèle simplifié de StarPU MPI $\rightsquigarrow$ isoler problèmes.
- Test StarPU: MPI, Cholesky $\rightsquigarrow$ valider modifications

Topic

1. Contexte et problématique
2. Évaluation des performances en HPC
3. Analyse du problème
4. Méthodologie
5. Contribution
6. Validation
7. Conclusion

Pour conclure

Bilan

- StarPU + SimGrid modifié pour simuler StarPU MPI.
- Difficulté: apporter modifications minimes dans un code non trivial. Environ 20 lignes sur un total de plus de 270 000

Prochaine étape

- Simulations et mesures avec solveur d'algèbre linéaire.
- Vérifications système réel: Grid5000.

Fin

Merci pour votre attention.