

TER

Simulation d'application dynamiques pour plateformes de
calculs hautes performances

Équipe MESCAL

Steven QUINITO MASNADA

Grenoble, 8 Juin 2015

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

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

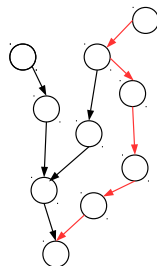
API de communication :

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

- Utiliser plusieurs paradigmes \leadsto 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 \leadsto 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** \leadsto système dynamique plus simple.



- Librairie StarPU:
 - Système **runtime**
 - basé sur le paradigme de tâches \leadsto graphe de dépendances.
 - Ordonnancement **dynamique et opportuniste**.
- **Problématique** : Performances difficiles à évaluer
 - Configuration **runtime**, heuristique, politique ordonnancement.
 - Configuration **application**, découpage des tâches.

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

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

Généralités

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

Simulation par rejeu de trace

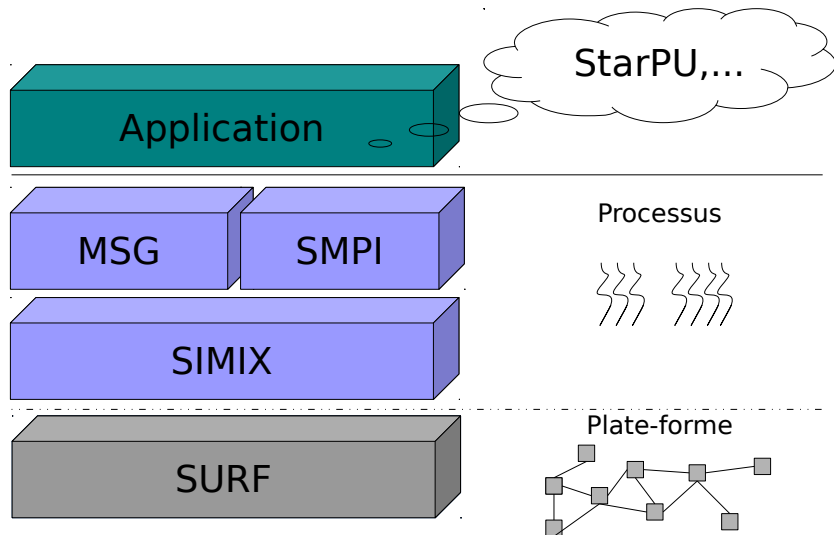
Exécution post-mortem: pas adapté ici car **flot de contrôle non déterministe**.

Hybride simulation / émulation

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

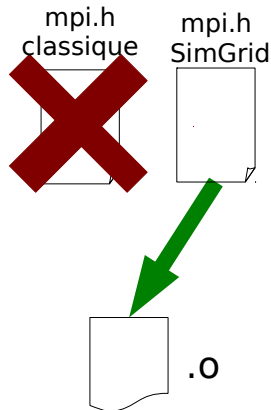
StarPU porté récemment au dessus de Simgrid.

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

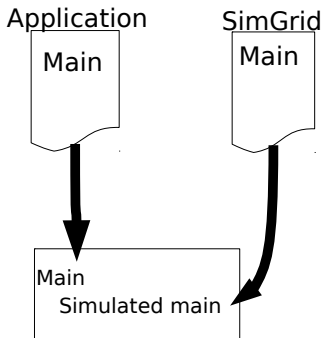


Construction de l'application MPI simulée

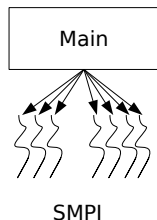
Compilation



Édition de liens

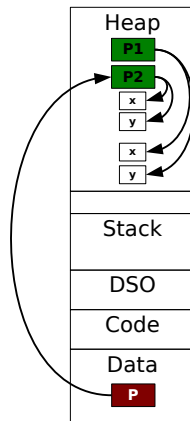


Exécution



Privatisation segment data

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

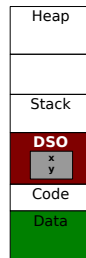


- Première version StarPU \leadsto hybride
- Basé sur MSG car modèle de performance plus proche (communications, environnement mémoire partagé), CPUs GPUs.
- Simulation: calculs, allocations mémoire des tâches, transfert CUDA.

StarPU SMPI: Difficultés de mise en oeuvre

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

- Problème des librairies dynamiques.



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

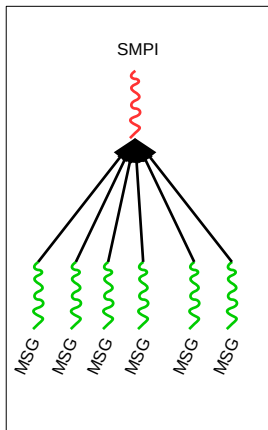
- Dépôt git submodules:
 - StarPU SMPI:
 - SimGrid
 - StarPU
- Suivi:
 - Cahier de laboratoire org mode github.
- Compréhension:
 - Documentation.
 - SimGrid = 106 350 lignes de codes.
 - StarPU = 172 251 lignes de codes.
 - "Code mining" et vérifications: GDB, Valgrind.

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

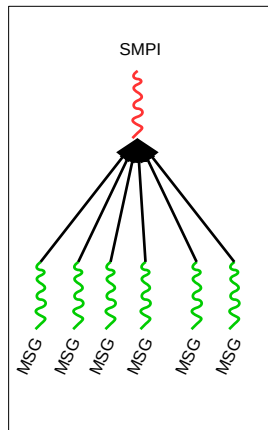
Modification de SimGrid

- Initialisation MSG + SMPI
- Gestion segment data

Noeud x

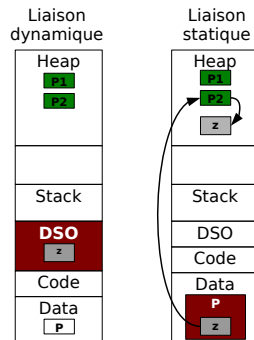


Noeud y



Librairie dynamiques et modifications StarPU

- Bibliothèques dynamiques:
 - Utilisation bibliothèques statiques.
- Modification StarPU:
 - Initialisation, car privatisation tardive.



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

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

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

Bilan

- StarPU + SimGrid modifié pour simuler StarPU MPI.
- Difficulté: apporter **modifications minimales** 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.