Ordonnancement et gestion mémoire en environnement hétérogène à grande échelle

Intégration dans l'équipe REALOPT

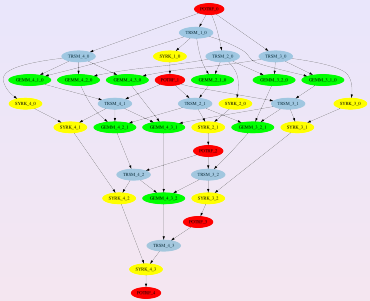
Julien Herrmann

- 2016 - 2018 : Postdoc
  Partitionnement de Graphe (heuristiques et implémentation), Ordonnancement
  The Ohio State University / Georgia Institute of Technology
  Ümit V. Çatalyürek
- 2012 - 2015 : Thèse
  Ordonnancement théorique, NP-complétude, Heuristiques, Algorithmes numériques
  ENS Lyon
  Y. Robert, L. Marchal

Contexte : Calcul Haute Performance (HPC)

Histoire

- Emergence de nouvelles unités de calculs et de nouveaux systèmes de stockage
- Architecture de plus en plus hiérarchiques et hétérogènes
- Nécessité de systèmes d'exécution plus portable
- Modélisation des applications sous forme de graphe de tâches



Tianhe-2

- 16, 000 noeuds
- 48, 000 Intel Xeon Phi
- 3, 120, 000 coeurs
- Numéro 2 du TOP500 (33.86 PetaFLOP/s)

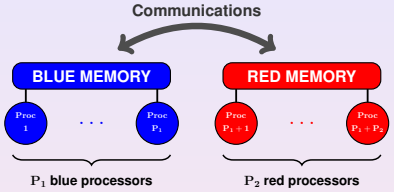
Contribution 1 : Ordonnancement de graphe de tâches avec deux types de mémoires

Modèle pour l'architecture

- Deux types de processeurs hétérogènes
- Deux mémoires dédiées
- Tâches hétérogènes avec de gros fichiers I/O

Modèle pour l'application

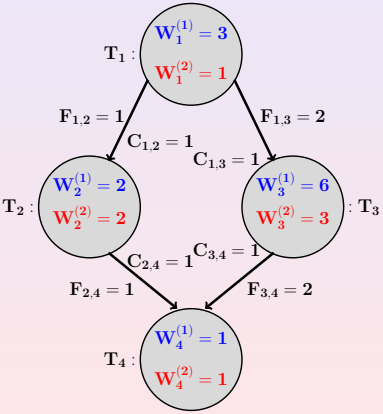
- Graphe de tâches avec des temps d'exécution différents
- Dépendances : fichiers avec des tailles différentes

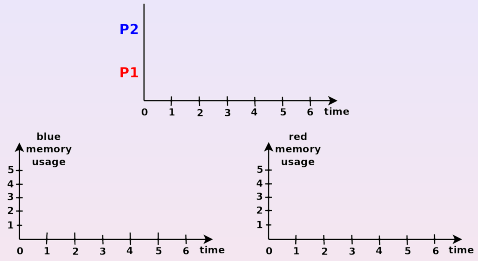


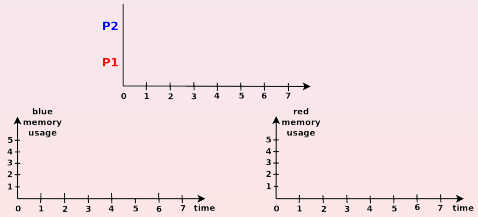
Exécution

- Lors de l'exécution d'un noeud, les fichiers d'entrée et de sortie doivent tenir en mémoire
- Après l'exécution d'un noeud son fichier d'entrée est effacé
- L'ordre d'exécution des noeuds impacte la consommation mémoire

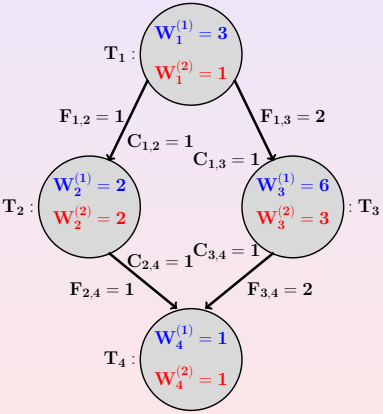
Contribution 1 : Ordonnancement de graphe de tâches avec deux types de mémoires

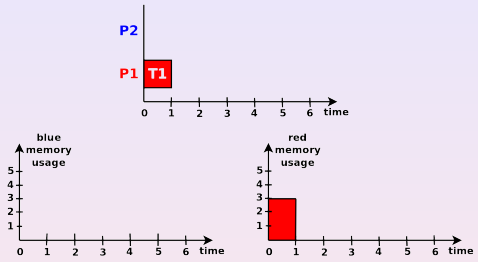


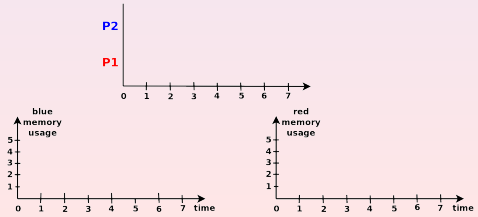




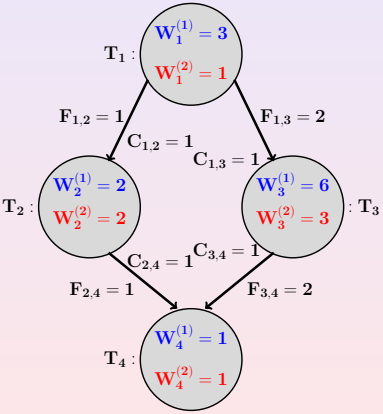
Contribution 1 : Ordonnancement de graphe de tâches avec deux types de mémoires

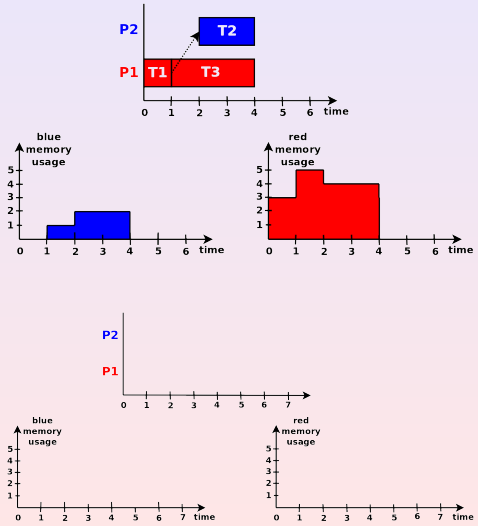




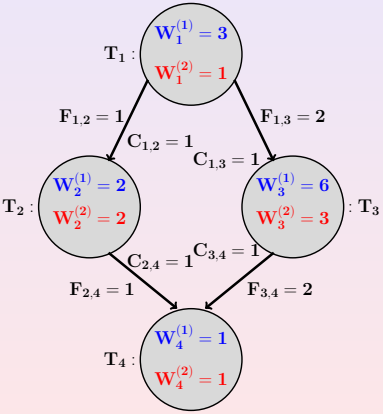


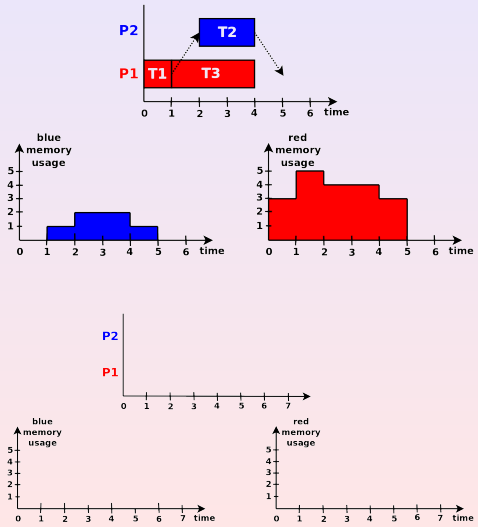
Contribution 1 : Ordonnancement de graphe de tâches avec deux types de mémoires



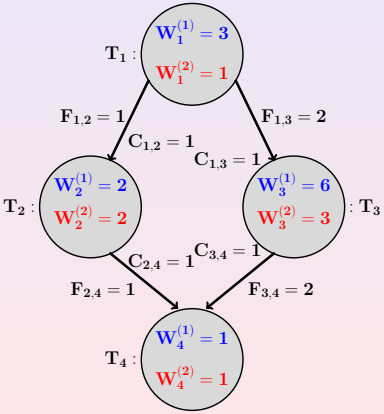


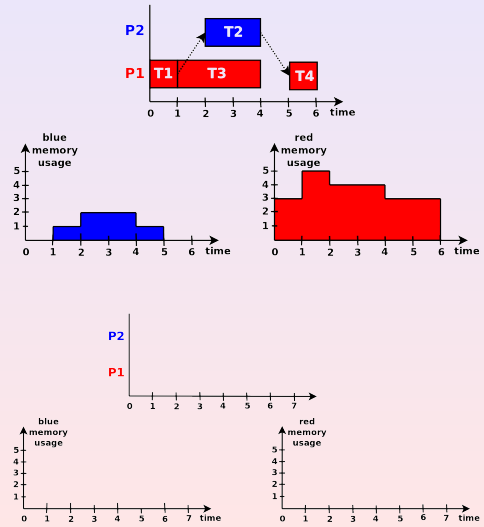
Contribution 1 : Ordonnancement de graphe de tâches avec deux types de mémoires



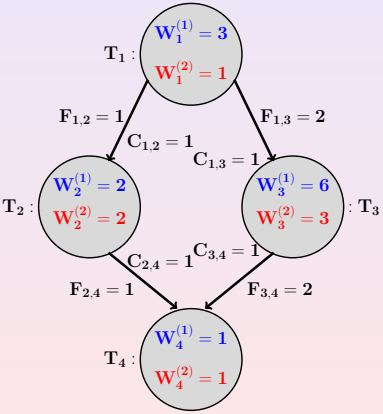


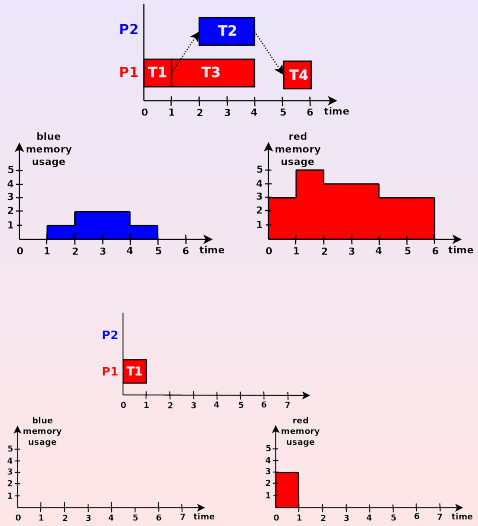
Contribution 1 : Ordonnancement de graphe de tâches avec deux types de mémoires



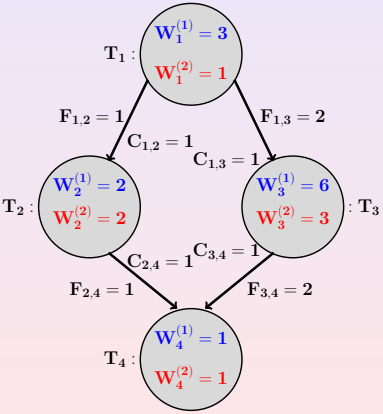


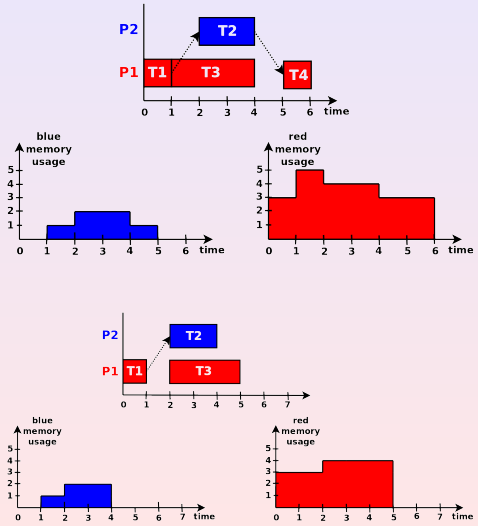
Contribution 1 : Ordonnancement de graphe de tâches avec deux types de mémoires



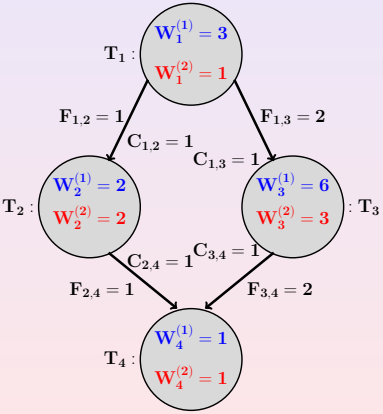


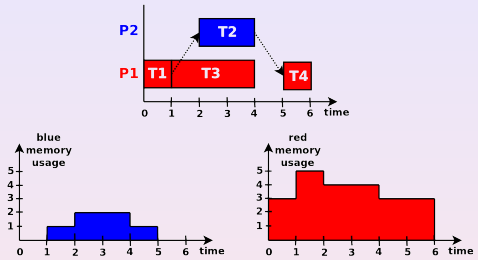
Contribution 1 : Ordonnancement de graphe de tâches avec deux types de mémoires

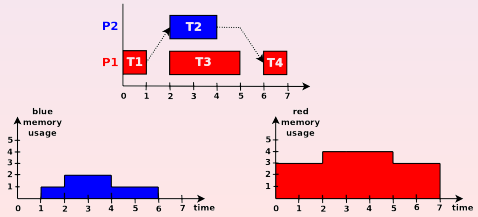




Contribution 1 : Ordonnancement de graphe de tâches avec deux types de mémoires



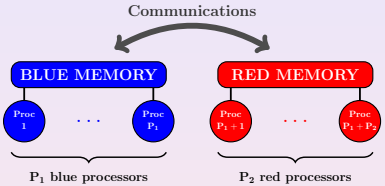




Contribution 1 : Ordonnancement de graphe de tâches avec deux types de mémoires

Graphes en forme d'arbres (allocation fixe) [Euro-Par 2013 + JPDC 2015]

- Traversée optimale avec une mémoire illimitée
- Traversée en profondeur d'abord optimale pour chaque mémoire
- Résultats d'inapproximabilité pour la minimisation des deux mémoires



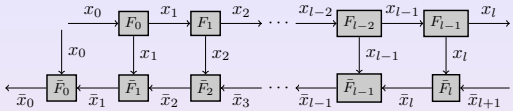
Graphes généraux [APDCM 2014]

- Formulation du programme linéaire
- Heuristiques pour la minimisation du makespan avec contraintes mémoire

Contribution 2 : Calcul auto-adjoint

Contexte

- Succession de différentiations :
  - Application Physiques
  - Machine Learning
  - Optimisations Mathématiques
  - Systèmes d'équations non-linéaires
- Deux mémoires :
  - In core : accès instantané mais taille limitée
  - Out-of-core : temps d'accès non négligeable mais taille illimitée



Collaborations

- Paul Hovland – Argonne National Laboratory
- Guillaume Aupy – ENS Lyon

Contributions [SISC 2015 + OMS 2017]

- Premier algorithme exact à résoudre le problème
- Problème ouvert depuis 2008

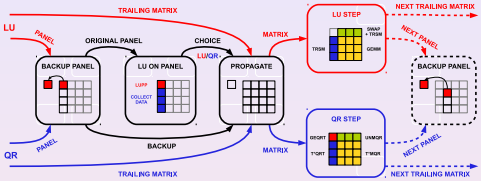
Contribution 3 : Performances d'applications HPC (au niveau des algorithmes numériques)

Résolution des systèmes linéaires

- Trouver x tel que $Ax = b$
- Factorisation LU : rapide
- Factorisation QR : stable numériquement

Contributions [IPDPS 2014, JPDC 2015]

- Solveur linéaire hybride LU / QR
- Trois critères de stabilité
- Implémentation en PaRSEC



Collaborations

- Jack Dongarra, Mathieu Faverge – University of Tennessee
- Julien Langou – University of Colorado

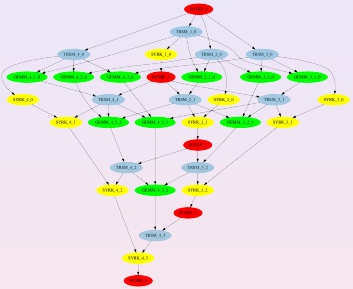
Contribution 3 : Performances d'applications HPC (au niveau de l'ordonnancement)

Contexte

- Factorisation Cholesky
- Plateformes hétérogènes : CPUs + GPUs
- Affinités par tâches

Contributions [HCW 2015]

- Amélioration de la borne supérieure à l'aide d'un problème de satisfaction de contraintes
- Analyse statique pour l'amélioration des ordonnanceurs dynamiques
- Implémentation en StarPU



Collaborations

- Olivier Beaumont, Lionel Eyraud-Dubois – INRIA Bordeaux

Contribution 4 : Partitionnement de graphes dirigés acycliques

Partitionnement de graphe

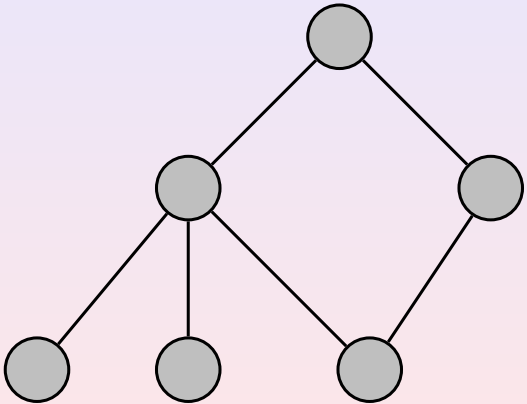
- Partitionner les noeuds du graphe en K parties de taille égales
- Minimiser le coût de coupe
- Problème NP-difficile

Partitionnement convexe

- Graphe dirigé acyclique
- Graphe de dépendance entre les partitions acyclique

Application

- Détecter la localité mémoire
- Allouer les graphes d'applications "pipelinée"



Contribution 4 : Partitionnement de graphes dirigés acycliques

Partitionnement de graphe

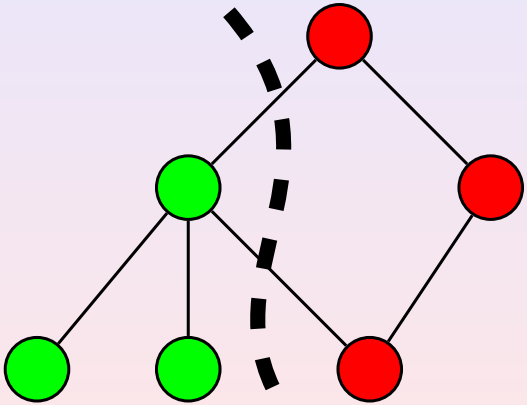
- Partitionner les noeuds du graphe en K parties de taille égales
- Minimiser le coût de coupe
- Problème NP-difficile

Partitionnement convexe

- Graphe dirigé acyclique
- Graphe de dépendance entre les partitions acyclique

Application

- Détecter la localité mémoire
- Allouer les graphes d'applications "pipelinée"



Contribution 4 : Partitionnement de graphes dirigés acycliques

Partitionnement de graphe

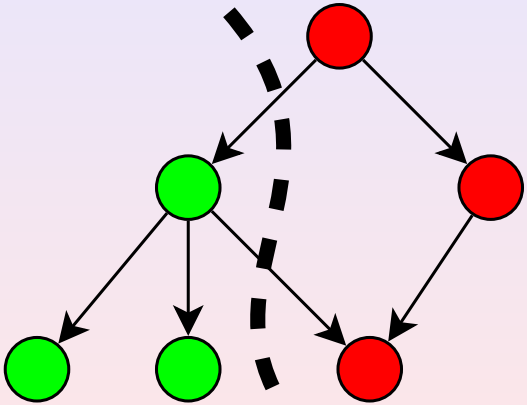
- Partitionner les noeuds du graphe en K parties de taille égales
- Minimiser le coût de coupe
- Problème NP-difficile

Partitionnement convexe

- Graphe dirigé acyclique
- Graphe de dépendance entre les partitions acyclique

Application

- Détecter la localité mémoire
- Allouer les graphes d'applications "pipelinée"



Contribution 4 : Partitionnement de graphes dirigés acycliques

Partitionnement de graphe

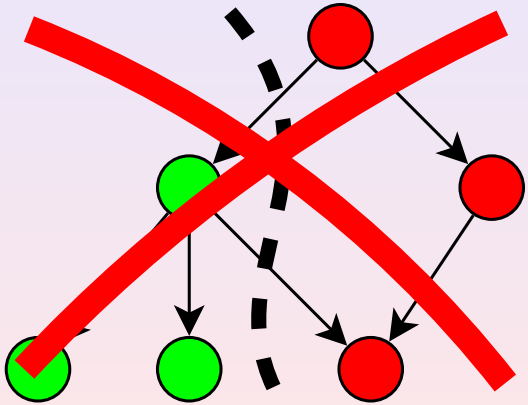
- Partitionner les noeuds du graphe en K parties de taille égales
- Minimiser le coût de coupe
- Problème NP-difficile

Partitionnement convexe

- Graphe dirigé acyclique
- Graphe de dépendance entre les partitions acyclique

Application

- Détecter la localité mémoire
- Allouer les graphes d'applications "pipelinée"



Contribution 4 : Partitionnement de graphes dirigés acycliques

Partitionnement de graphe

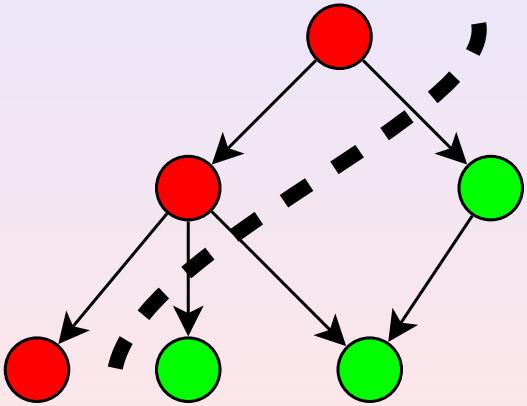
- Partitionner les noeuds du graphe en K parties de taille égales
- Minimiser le coût de coupe
- Problème NP-difficile

Partitionnement convexe

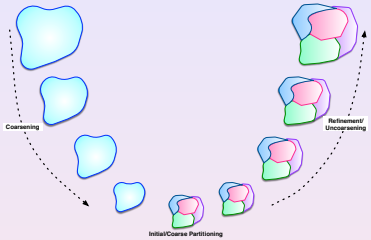
- Graphe dirigé acyclique
- Graphe de dépendance entre les partitions acyclique

Application

- Détecter la localité mémoire
- Allouer les graphes d'applications "pipelinée"



Contribution 4 : Partitionnement de graphes dirigés acycliques



Contribution [CCGrid 2017 + SISC]

- Partitionneur convexe multi-niveaux
- Nouvelles heuristiques de regroupement et de raffinement pour le cas convexe
- Implémentation dans une librairie C / C++ (14,000 lignes de code)

Travaux en cours

- Utilisation du partitionneur convexe pour le calcul de traversée de graphe minimisant l'utilisation mémoire
- Utilisation du partitionneur convexe pour l'ordonnancement basé sur le partitionnement
- Intégration dans l'environnement OCR

Collaborations

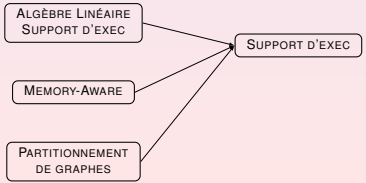
- Ümit V. Çatalyürek – Georgia Institute of Technology

Projet 1 : Support d'exécution dynamiques pour architectures hétérogènes à grande échelle

Contexte

- L'hétérogénéité des architectures ne va faire qu'augmenter (Xeon Phi, TPU,...)
- Architecture de plus en plus hiérarchiques
- Communications et accès mémoires de plus en plus critiques
- Nécessité d'adapter les systèmes d'exécution aux défis Exascale
  ↪ Projet ANR SOLHARIS (Hiepacs, RealOpt, Storm, Tadaam, ...)

Mon Expérience **Mon Projet**



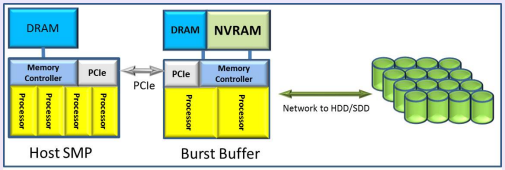
Projet

- Conception de nouveaux algorithmes minimisant les communications
  ↪ Expérience dans les algorithmes LU-QR
  ↪ Étendre aux mix LU-LU
  ↪ Étendre à l'algèbre linéaire creuse
- Redistribuer ou répliquer les données avant l'exécution
- Bon partitionnement des données lors de l'exécution
  ↪ Expérience dans les algorithmes de partionnement et dans la manipulation de gros graphes

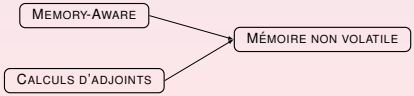
Projet 2 : Stockage mémoire non volatile

Contexte

- Émergence des Burst Buffers
- Mémoire intermédiaire avec une grosse bande passante d'entrée
- Absorber les pics d'écriture rapidement
- Tianhe-2 et Summit
  ↪ Projet de recherche BCube (RealOpt, Tadaam, ...)



Mon Expérience **Mon Projet**



Projet

- Analyser le comportement des Burst Buffers
  ↪ Travaux récents par RealOpt et TADaaM
- Étendre à un model plus complexe (prefetching, Burst Buffers distribués...)
- Étude de la gestion des données dans le Burst Buffers
  ↪ Expérience dans l'étude des graphes auto-adjoint

Projet 3 : Stratégies d'ordonnancement pour le Deep Learning haute performance

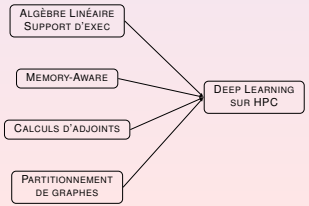
Calcul Haute Performance

- Les simulations génèrent de plus en plus de données
- Les pannes obligent à stocker des checkpoints
- Ajout de gros stockage plus près des noeuds

Deep Learning

- Calculs intensifs
- Rétro-propagation sous forme d'auto-adjoint
- Nécessité d'optimisation de l'allocation de ressources et ordonnancement

Mon Expérience **Mon Projet**



Convergence Big Data - HPC [Projet IPL]

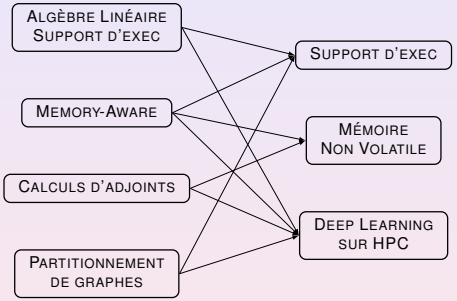
- Parallelisme de la phase d'apprentissage
- Analyse statique des applications
  ↪ Expérience en ordonnancement de Cholesky
  ↪ Expérience dans l'analyse de sujets de recherche éloignés de mon domaine
- Projet IPL "Convergence Big Data - HPC" (RealOpt, Zenith, ...)

Conclusion

- 24 co-auteurs (14 appartenant à des institutions étrangères : Univ. of Tennessee, Georgia Tech, Univ. of Hawaii, Argonne Nat Lab, UNC Charlotte, Univ. of Denver)
- 3+ mois en séjour de recherche durant la thèse
- Aide à l'encadrement de stagiaires / doctorants
- Diversification de mes recherches

- 14 articles (8 confs, 6 journaux)
- 2 comités locaux d'organisation
- 2 ans élu au Conseil du labo
- Diffusion mathématiques (Plaisir-Maths)

Mon Expérience **Mon Projet**



Nombreux développements technologiques

- Librairie C++ de partitionnement convexe de graphes dirigés acycliques (14,000 lignes de code)
- Techniques de redistribution des données suivie par un noyaux de calcul dans PaRSEC
- Techniques d'ordonnancement dynamiques guidées par des informations statiques dans StarPU
- Solveur hybride de systèmes linéaires alternant étapes de factorisation LU et QR dans PaRSEC