**Résumé**

**Ordonnancement d'un DAG avec transfert des données sur une plateforme NUMA multicore**

Les algorithmes d'ordonnancement des graphes de tâches (DAG) selon l'approche classique se concentrent davantage sur les tâches de calcul et leurs dépendances (communication) et donnent moins d'importance au placement des données lors de l'allocation. Dans le contexte des plateformes hiérarchiques, ne pas placer les données des threads de façon appropriée impacte les performances du système. Avec l'introduction des architectures d'accès non uniforme à la mémoire (NUMA), l'optimisation simultanée du placement des threads et des données d'un DAG est devenue cruciale. Une telle architecture possède un système de mémoire hiérarchique multi niveaux avec des caractéristiques différentes qui rend le coût de la communication différent dans chaque niveau. Une bonne partie du temps des tâches dépendantes orientées données se consomme en chargeant ou sauvegardant ses données de la mémoire en fonction du placement initial. Les performances d'un tel système et son ordonnanceur dépendent non seulement de sa décision d'affectation des threads mais aussi de sa décision du placement des données.

Dans cette thèse, nous abordons le problème de l'ordonnancement d'applications parallèles décrites par un DAG sur la plateforme cible en explorant le cas où non seulement le calcul et la communication sont considérés, mais aussi le placement de données sur les plateformes hiérarchiques. Les politiques courantes de l'ordonnancement et l'allocation de la mémoire tentent d'atténuer l'eet NUMA en réduisant la pénalité d'accès distant à la mémoire en tenant en compte la localité des données lors de la décision de l'ordonnancement. Ce travail propose de :

1. Inclure l'information sur la topologie (fournie par l'environnement runtime au début de l'exécution de l'application) de la plateforme.

2. Utiliser le motif de l'application (sa classe information fournie par le développeur de l'application) à exécuter.

3. Regrouper les tâches dans des classes disjointes en se basant sur cette structure et l'état de la tâche.

4. Elargir le champs de visibilité des tâches en explorant un large horizon an de collecter plus d'informations sur l'état courant du processus d'exécution de DAG.

5. Equilibrer la charge au temps d'exécution en utilisant la stratégie de vol de travail basée sur la distance.

Comme idées de bases, ces propositions seront intégrées comme heuristiques dans la politique de l'ordonnancement des threads et le placement des données pour guider ce processus et l'aider à réduire l'eet NUMA sur le temps total d'exécution d'un DAG et de préserver les performances du système.

Mots-clés : Ordonnancement DAG, Placement, Localité de données, Machines multicoeurs, Architectures NUMA, Caches hiérarchiques.