Ordonnancement pour la gestion de la mémoire et du préchargement dans les architectures multicoeurs embarquées

Sergiu Carpov

Soutenance de thèse de doctorat

Institut de Physique Théorique, Orme des Merisiers

14 octobre 2011

Parcours

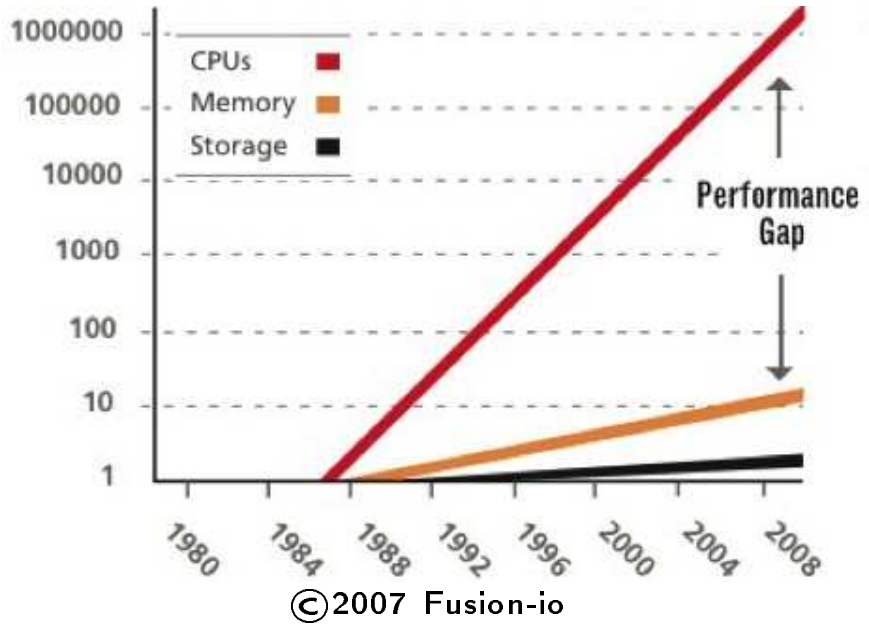
- 2007 : diplôme d'ingénieur en informatique de l'Université Technique de Moldavie.
  - 2006 à 2007 : ingénieur développement logiciel, compagnie « Endava ».
- 2008 : diplôme Master 2 Recherche de l'Université de Technologie de Compiègne. Spécialité : Technologies de l'Information et des Systèmes.
  - Stage au sein du laboratoire LaSTRE du CEA Saclay.
- Depuis octobre 2008 : préparation d'une thèse de doctorat dirigée par MM. J. Carlier et D. Nace (Heudiasyc/UTC), encadré par M. R. Sirdey (LaSTRE/CEA).

Table des matières

Compilation pour architectures multicoeurs

Problématique de la gestion de la mémoire

- « Nouvelle loi de Moore »
  - Le nombre de ~~transistors~~ coeurs double tous les 2 ans
- Écart de performance entre les cœurs de calcul et l'accès à la mémoire
  - Constat exacerbé pour les multicoeurs
- Mémoire à plusieurs niveaux (hiérarchies mémoire)
- Optimisation des accès à la mémoire :
  - Réutilisation des données
  - Préchargement (spéculatif) des données
- Étude de ces techniques pour les multicoeurs



Réutilisation vs préchargement des données

Réutilisation des données

Stocker certaines données dans une mémoire intermédiaire (e.g., un cache) pour une réutilisation ultérieure

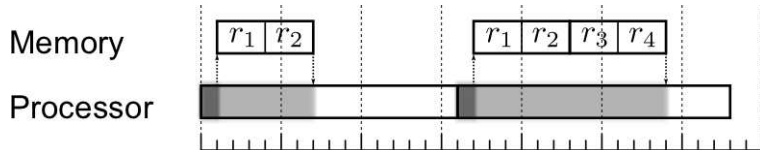
Préchargement des données

Charger des données avant d'en avoir besoin

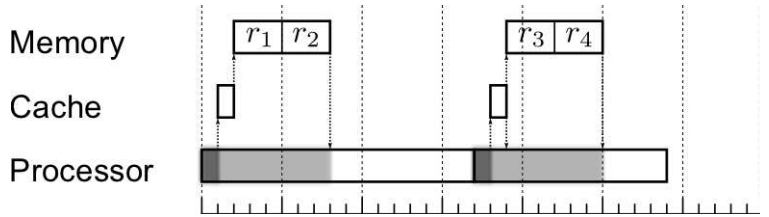
Préchargement spéculatif

Charger des données avant de savoir si l'on va en avoir besoin

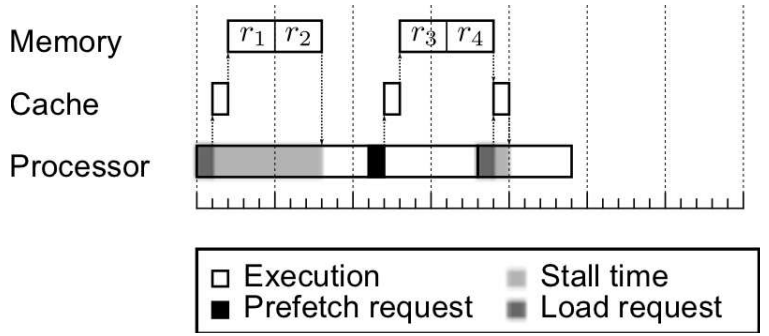
~~Réutilisation~~ ~~Préchargement~~



Réutilisation ~~Préchargement~~

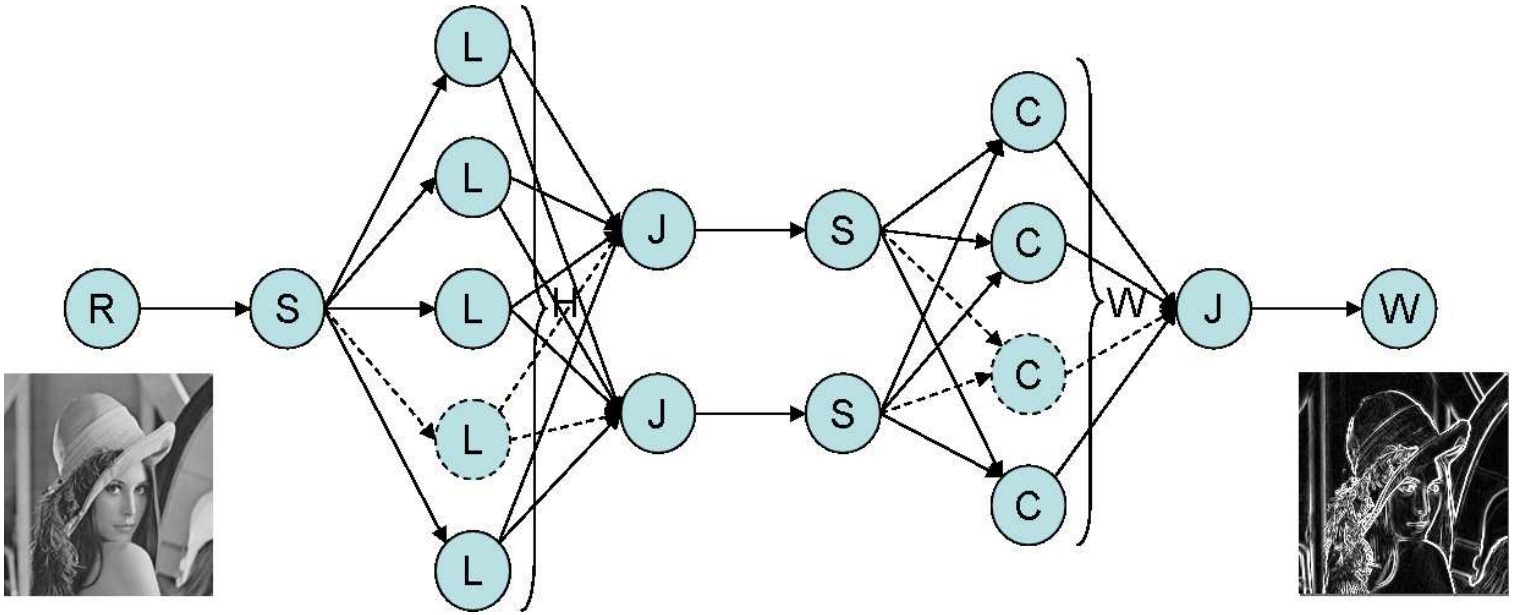


Réutilisation Préchargement



La programmation orientée flot de données

- Difficultés de la programmation parallèle : parallélisme, déterminisme et accessibilité
- Les applications flot de données sont des réseaux de tâches communicant à travers des canaux FIFO
- Principaux modèles : Synchronous DataFlow, Cyclo-static DataFlow, Boolean DataFlow, Integer DataFlow



Cadre applicatif potentiel

- Architecture massivement multicoeur clusterisée MPPA
- Langage ΣC : extension avec contrôle du modèle CSDF :
  - *Bénigne en termes de décidabilité*
  - Conçu et développé par le laboratoire pour la puce MPPA
- Modèle d'exécution cadencé par un temps logique :
  - Cadre formelle suffisamment riche pour l'exécution performante d'applications flot de données avec contrôle
  - Conçu et breveté par le laboratoire pour la puce MPPA
- Problèmes d'optimisation représentatifs pour la compilation d'une application ΣC :
  - Partitionnement de graphes
  - Problème d'affectation quadratique
  - Multi-flot sous contraintes
  - Ordonnancement partiel sous contraintes

État de l'art

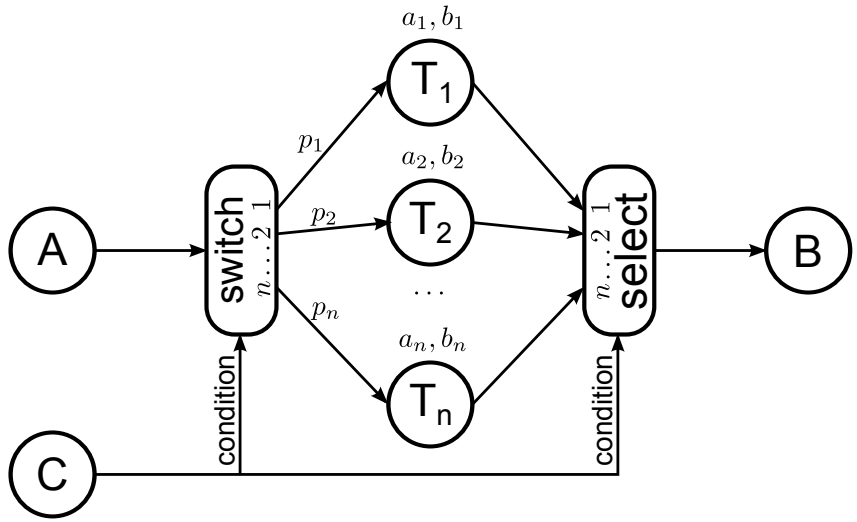
- Préchargement pour les monocœurs :
  - Approches matériel et/ou logiciel
  - Bons résultats pour les programmes ayant des accès réguliers à la mémoire
- Préchargement pour les architectures parallèles :
  - Peu étudié
  - Difficile d'utiliser des techniques monocœur, car il faut partager la bande passante mémoire (pas de préchargement agressif)
  - Utilisation des processus d'exécution auxiliaires pour cacher la latence de chargement des données :
    - Exécution préalable
    - Prédiction des accès futurs
- Pas de travaux identifiés dans le cadre des langages de programmation orientés flot de données

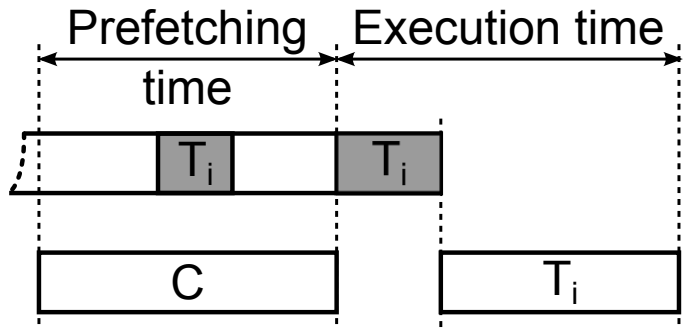
Préchargement spéculatif pour une structure de branchement

Structure de branchement - préchargement spéculatif

Structure de branchement - permet une exécution conditionnelle dans une application flot de données (switch, select)

- Une branche T_i désignée par la « condition » est exécutée
- Deux étapes par branche :
  - Chargement données - durée a_i
  - Exécution - durée b_i
- Temps de préchargement T - une variable aléatoire
- Minimisation du temps d'exécution
- Stratégies de préchargement :
  - Fractionnaire
  - Tout-ou-rien (privilégiée)





Carpov et al., ISCO 2010

Minimisation de l'espérance du temps d'exécution (1)

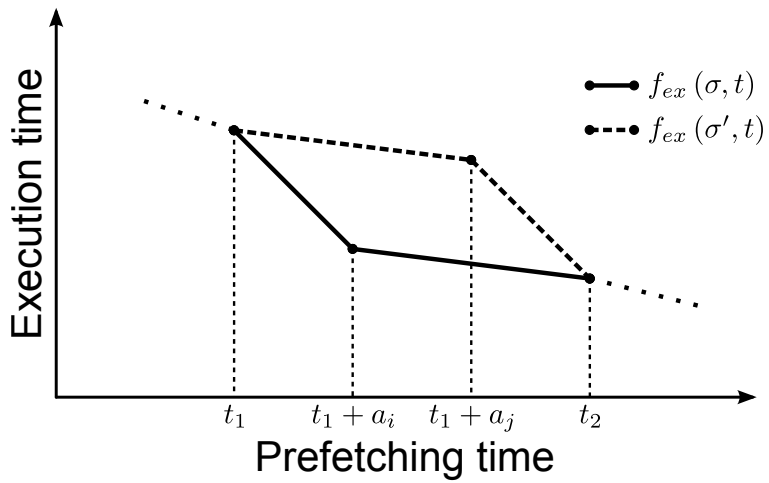
- Probabilité d'exécution d'une branche p_i
- $f_{ex}(\sigma, t)$ - espérance mathématique du temps d'exécution pour un ordre σ de préchargement, temps t de préchargement
  - Ex. $f_{ex}(.,0) = \sum_i p_i (a_i + b_i)$

Objectif

Trouver σ pour que $f_{ex}(\sigma, T)$ soit minimal pour un temps de préchargement aléatoire T

Stratégie optimale

Précharger les branches dans l'ordre décroissant des probabilités tant que le temps de préchargement le permet



Minimisation de l'espérance du temps d'exécution (2)

- Modélisation du problème de préchargement fractionnaire sous la forme d'un sac-à-dos continu
- Algorithme de Dantzig polynomial
  - Préchargement optimal pour la stratégie tout-ou-rien

Préchargement fractionnaire

$$\text{Minimize} \quad \sum_i p_i \left(a_i - \alpha_i + b_i\right)$$

$$\text{s.t.} \quad \sum_i \alpha_i = t$$

$$\alpha_i \in [0, a_i], \ \forall i$$

Sac-à-dos continu

$$\text{Maximize} \quad \sum_i p_i a_i x_i$$

$$\text{s.t.} \quad \sum_i a_i x_i = t$$

$$x_i \in [0, 1], \ \forall i$$

Minimisation du pire temps d'exécution (1)

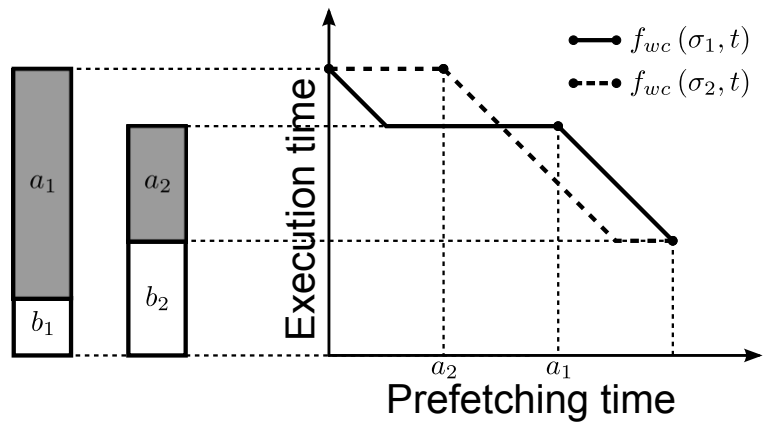
- $f_{wc}(\sigma, t)$ - pire temps d'exécution pour un ordre de préchargement des branches σ et un temps de préchargement t
- Un ordre σ qui minimise $f_{wc}(\sigma, t)$ ne peut pas être toujours défini indépendamment de t

Objectif

Trouver σ pour que $E[f_{wc}(\sigma, T)]$ soit minimal pour un temps de préchargement aléatoire T

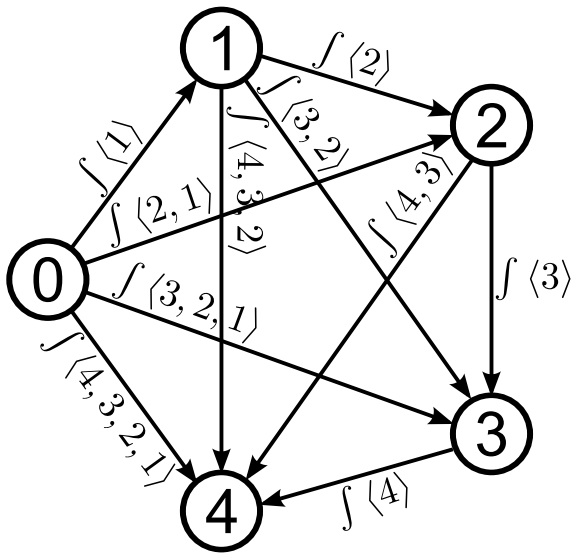
Stratégie optimale

Résolution polynomiale par recherche de plus court chemin dans un graphe particulier



Minimisation du pire temps d'exécution (2)

- Stratégie optimale basée sur deux propriétés de dominance entre les ordres de préchargement



- Algorithme de programmation dynamique

Calcul de l'ordre optimal σ_n de préchargement

1: $c_0 = 0$
2: $c_i = \infty$ for any $i = 1, \ldots, n$
3: $\sigma_i = \emptyset$ for any $i = 0, \ldots, n$
4: **for** $i = 1$ **to** n **do**
5: **for** $j = 0$ **to** $i - 1$ **do**
6: $\sigma = \sigma_j + \langle i, i-1, \ldots, j+1 \rangle$
7: $c = \int_{l_j}^{l_i} f(\sigma, t)\, dt$
8: **if** $c_i > c_j + c$ **then**
9: $c_i = c$
10: $\sigma_i = \sigma$
11: **end if**
12: **end for**
13: **end for**

Préchargement pour un ordre partiel

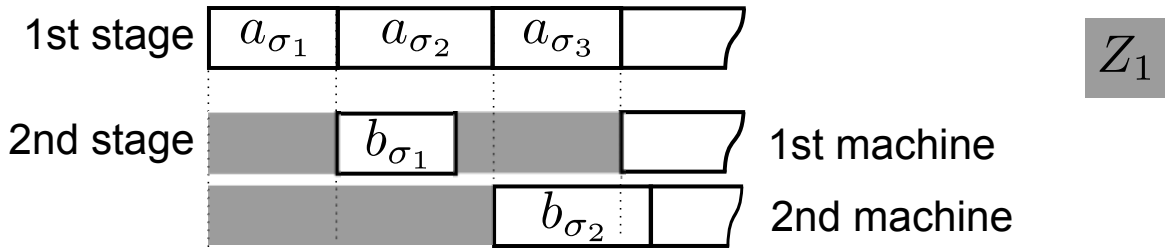
Flow shop hybride avec relations de précédence

- Approximation à horizon fini pour le problème de préchargement d'un ordre partiel
- Chaque tâche comprend deux opérations :
  - Chargement des données - exécutée sur une machine (le canal d'accès à la mémoire), durée a_i
  - Exécution proprement dite - exécutée sur une des m machines (les processeurs parallèles), durée b_i
  - Avec (« classic ») ou sans (« no-wait ») temps mort entre les étapes
- Relations de précédence entre les opérations d'exécution
- Objectif - minimisation du temps total d'exécution
- Problème $\mathcal{NP}$-difficile au sens fort
- Méthode de résolution : heuristique + borne inférieure globale

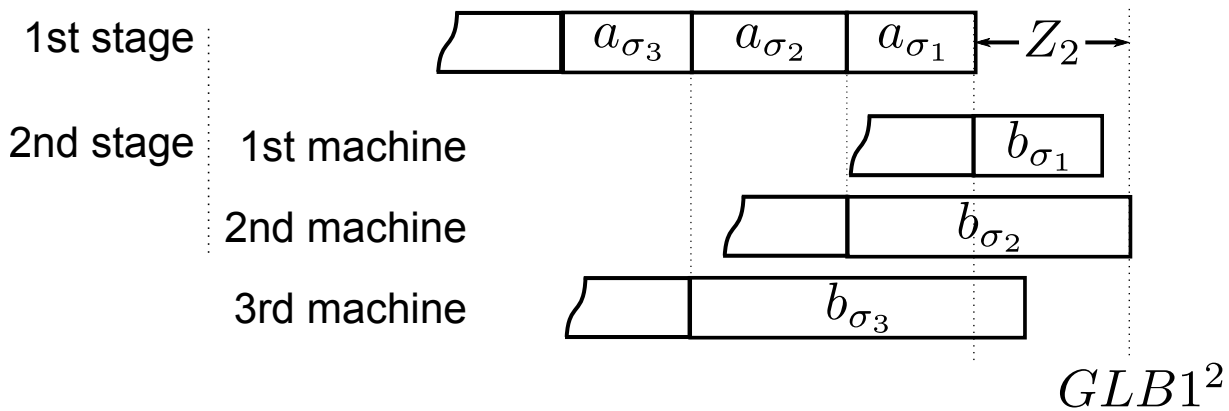
Carpov et al., COR 2012

Borne 1

- Première borne inférieure globale : $GLB1 = \max\left(GLB1^1, GLB1^2\right)$
- $GLB1^1 = \left\lceil \frac{1}{m}\left(Z_1 + \sum_{i=1}^{n} b_i\right)\right\rceil$, où Z_1 dénote le temps mort sur la deuxième étape :



- $GLB1^2 = \sum_{i=1}^{n} a_i + Z_2$, où Z_2 dénote le plus petit dépassement :



Borne 2

- Ajustement des « heads » et des « tails » pour chaque tâche :
  - Head - la date de disponibilité, r_i^I, r_i^{II}
  - Tail - la durée de latence, q_i^I, q_i^{II}
- Deuxième borne inférieure globale : $GLB2 = \max_i \left(r_i^{II} + b_i + q_i^{II}\right)$
- Modèle de programmation par contraintes
- Contraintes utilisées :
  - Relations de précédence entre les étapes et entre les tâches
  - Travail antérieur cumulé
  - Contraintes énergétiques
  - Règle de Jackson préemptive
- Propagation de contraintes + dichotomie
  - Méthode pseudo-polynomiale

Résolution heuristique

- Résolution approchée - algorithme de liste adaptatif randomisé
  - Algorithme de liste adaptatif randomisé, 2 étapes :
    - Échantillonage pour la fixation du paramètre de randomisation
    - Algorithme de liste exécuté $O\left(n^k\right)$ fois, $k \in [1.5, 2]$
  - Garde la meilleure solution obtenue
  - Deux fonctions de priorité
- Instances - générées aléatoirement à partir du Standard Task Graph set
  - 50 et 100 tâches
  - Durée des tâches - lois exponentielle, normale et uniforme
  - Répartition de charges - 50% plus de chargement, 50% plus d'exécution et équilibrée
  - De 2 à 10 machines sur la deuxième étape
  - 9720 instances ont été générées

Résultats expérimentaux

- Pour chaque instance une solution approchée et deux bornes sont calculées
- Borne $GLB2$ meilleure pour plus de 60% d'instances
- Qualité de la borne $GLB1$ augmente avec la taille du problème (de 8.5% à 12.6%)
- Déviations moyennes des solutions ARLS par rapport à l'optimum :
  - Classique inférieure à 2%
  - No-wait inférieure à 5%

Déviations moyennes des solutions par rapport aux bornes

	50		100	
	LS	ARLS	LS	ARLS
Classic	2.98%	1.83%	1.95%	1.21%
No-wait	7.83%	4.62%	7.97%	4.94%

Optimisation de la réutilisation mémoire

Séquencement de tâches et gestion de la mémoire (1)

- Séquencer un ensemble de tâches pour maximiser la réutilisation des données
  - Ensemble des tâches $s_1, \ldots, s_n$
  - $\delta(s_i)$ données utilisées par s_i
- Les données sont soit chargées depuis la mémoire externe ou réutilisées depuis une mémoire interne (petite taille)
  - $\gamma(s_i)$ état de la mémoire interne avant l'exécution de s_i
  - $\delta(s_i) \subseteq \gamma(s_i)$
- Trouver un ordre π d'exécution des tâches afin de minimiser le nombre des accès à la mémoire externe :

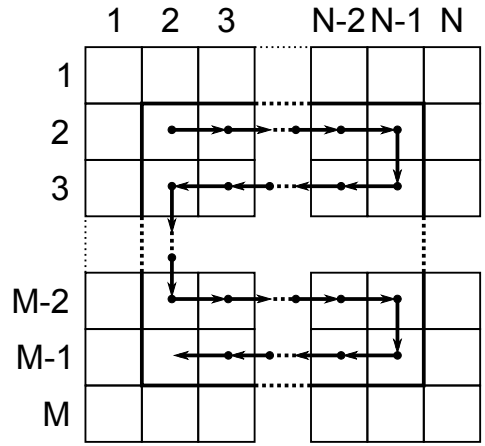
$$\min_\pi \left[\left| \gamma\left(s_{\pi(1)}\right) \right| + \sum_{i=2}^{|S|} \left| \gamma\left(s_{\pi(i)}\right) \setminus \gamma\left(s_{\pi(i-1)}\right) \right| \right]$$

- Problème $\mathcal{NP}$-difficile

Carpov et al., COCOON 2011

Séquencement de tâches et gestion de la mémoire (2)

- Méthode arborescente (branch and bound) proposée
  - Deux bornes inférieures et une relation de dominance
- Deux heuristiques : 2STSP, RCI



- Évaluation sur des instances générées aléatoirement (5832 au total)
- 3854 des instances (approx. 67%) ont été résolues à l'optimalité
- Heuristique RCI surpasse 2STSP dans la majorité des cas
  - 83% des solutions optimales retrouvées par le RCI contre 66% par le 2STSP
  - Solutions RCI à moins de 2% des solutions optimales

Conclusions et perspectives

Conclusions

- Trois problèmes d'optimisation des accès à la mémoire
- Préchargement spéculatif pour une structure de branchement
  - Résolution optimale en temps polynomial
- Préchargement pour une application orientée flot de données
  - Introduction et étude du problème de flow shop hybride avec relations de précédence
  - Heuristique + bornes inférieures
- Séquencement de tâches et gestion de la mémoire
  - Résolutions exacte et heuristique

Perspectives

- Problèmes d'ordonnancement pour le préchargement avec durées probabilistes et exécution conditionnelle
  - Première tentative de calcul des heads et tails probabilistes [3]
  - Extension du flow shop hybride
    - *Flow shop hybride probabiliste ?*
- Considérer des arborescences de structures de branchement
- Étudier des objectifs de type dispersion dans le cadre des structures de branchement
- Prise en compte du caractère cyclique de certains modèles sans contrôle (SDF, CSDF)
- Valorisation des travaux dans la chaîne de compilation développée par le LIST pour la puce MPPA de Kalray (notamment)
- Application du préchargement spéculatif : FFT dynamique, H.265, suivi de cible, radio cognitive, etc.

Publications

[1] Sergiu Carpov, Jacques Carlier, Dritan Nace, and Renaud Sirdey. Memory bandwidth-constrained parallelism dimensioning for embedded many-core microprocessors. CPAIOR workshop on Combinatorial Optimization for Embedded System Design, 2010.

[2] Sergiu Carpov, Renaud Sirdey, Jacques Carlier, and Dritan Nace. Speculative data prefetching for branching structures in dataflow programs. *Electronic Notes in Discrete Mathematics*, 36 :119–126, 2010.

[3] Sergiu Carpov, Jacques Carlier, Dritan Nace, and Renaud Sirdey. Probabilistic parameters of conditional task graphs. In *Proceedings of the 5th International Workshop on Advanced Distributed and Parallel Network Applications*, 2011.

[4] Sergiu Carpov, Jacques Carlier, Dritan Nace, and Renaud Sirdey. Task ordering and memory management problem for degree of parallelism estimation. In *Lecture Notes in Computer Science*, volume 6842, pages 592–603, 2011.

[5] Sergiu Carpov, Jacques Carlier, Dritan Nace, and Renaud Sirdey. Two-stage hybrid flow shop with precedence constraints and parallel machines at second stage. *Computers & Operations Research*, 39(3) :736 – 745, 2012.

Merci !