

Prédiction de comportement d'applications parallèles et placement à l'aide de modèles économiques sur une grille de calcul

Soutenance de thèse en vue de l'obtention du
Doctorat de l'Université de Toulouse

Bernard Miegemolle

Directeur de thèse : Thierry Monteil

LAAS-CNRS

11 septembre 2008

Les grilles de calcul

Naissance du concept de grilles

- Augmentation des besoins en puissance de calcul
 - Calcul distribué
 - *Clustering*
- Avénement des réseaux longues distances à haut débit
 - Interconnexion de *clusters*
 - Grilles de calcul

Les grilles de calcul

Objectifs principaux

- Fournir une importante capacité de calcul parallèle
- Exploiter les ressources sous-utilisées
- Accéder à des ressources additionnelles
- Assurer une tolérance aux fautes pour un coût moindre

Les grilles de calcul

Objectifs principaux

- Fournir une importante capacité de calcul parallèle
- Exploiter les ressources sous-utilisées
- Accéder à des ressources additionnelles
- Assurer une tolérance aux fautes pour un coût moindre

Exemples de grilles

- SETI@home
- Grid'5000

Objectifs de la thèse

- Proposer un **modèle d'ordonnancement** permettant de :
 - partager équitablement l'accès aux ressources
 - réguler la demande sur les ressources de manière naturelle
- Proposer une **méthode de prédiction du temps d'exécution** des applications
 - automatiser l'estimation du temps d'exécution des applications
 - minimiser les sources d'erreurs

Plan de la présentation

- 1 Estimation du temps d'exécution d'une application
- 2 Résolution du modèle de prédiction hybride
- 3 Gestion des ressources d'une grille à l'aide d'un modèle économique
- 4 Mise en place d'un gestionnaire de ressources
- 5 Conclusion et perspectives

Plan de la présentation

- 1 Estimation du temps d'exécution d'une application
 - Sources d'inspiration
 - Méthode hybride de prédiction
- 2 Résolution du modèle de prédiction hybride
- 3 Gestion des ressources d'une grille à l'aide d'un modèle économique
- 4 Mise en place d'un gestionnaire de ressources
- 5 Conclusion et perspectives

Estimation du temps d'exécution d'une application

Sources d'inspiration

- Prédiction basée sur un historique d'exécutions passées
 ↪ *historic-based prediction*
- Prédiction basée sur le profil des applications
 ↪ *profile-based prediction*

Estimation du temps d'exécution d'une application

Estimation du WCET d'une application

Definition

WCET = *Worst-Case Execution Time*

Majorant des temps d'exécution d'un programme

- Domaine du temps réel :
 - Dimensionnement de systèmes temps réel
 - Détermination d'ordonnancements hors-lignes
 - Optimisation du code d'applications

Estimation du temps d'exécution d'une application

Estimation du WCET d'une application

- Méthodes dynamiques d'analyse du WCET
 - Mesure du temps d'exécution du programme
 - Principale difficulté : trouver un jeu d'entrées amenant au WCET de l'application
- Méthodes statiques d'analyse du WCET
 - 1 Analyse de flot
 - 2 Analyse de bas niveau
 - 3 Calcul du WCET

Estimation du temps d'exécution d'une application

Estimation du WCET d'une application

- Exemple d'analyse statique du WCET

```
void matrixMultiplication() {  
    for ( int i = 0; i < dimensions; i++ ) {  
        for ( int j = 0; j < dimensions; j++ ) {  
            resultMatrix[i][j] = 0;  
            for ( int k = 0; k < dimensions; k++ ) {  
                resultMatrix[i][j] += leftMatrix[i][k] * rightMatrix[k][j] ;  
            }  
        }  
    }  
}
```

Estimation du temps d'exécution d'une application

Estimation du WCET d'une application


- Exemple d'analyse statique du WCET

- 1. Analyse de flot

- Découper le programme en blocs élémentaires
 - Déterminer l'ensemble des chemins d'exécution

Definition

Bloc de base : séquence maximale d'instructions possédant un et un seul point d'entrée ainsi qu'un et un seul point de sortie dans le flot de contrôle du programme.



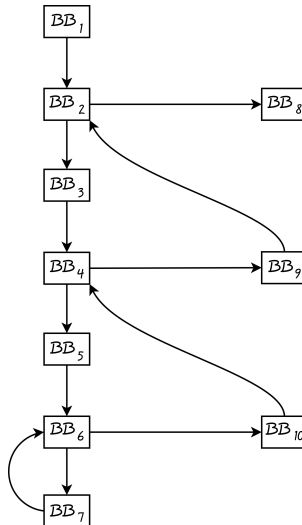
```
int i = 17;  
int j = 5;  
int k = i + 31 * j;
```

```

void matrixMultiplication() {
    for ( int i = 0; i < dimensions; i++ ) {
        for ( int j = 0; j < dimensions; j++ ) {
            resultMatrix[i][j] = 0;
            for ( int k = 0; k < dimensions; k++ ) {
                resultMatrix[i][j] += leftMatrix[i][k] * rightMatrix[k][j] ;
            }
        }
    }
}

```

Blocs de base	Code associé
BB_1	int i = 0;
BB_2	i < dimensions
BB_3	int j = 0;
BB_4	j < dimensions
BB_5	resultMatrix[i][j] = 0; int k = 0;
BB_6	k < dimensions
BB_7	resultMatrix[i][j] += leftMatrix[i][k] * rightMatrix[k][j] ; k++;
BB_8	Fin de la fonction
BB_9	i++;
BB_{10}	j++;



Estimation du temps d'exécution d'une application

Estimation du WCET d'une application

- Exemple d'analyse statique du WCET
 - **2. Analyse de bas niveau**
 - Déterminer le temps d'exécution des blocs de base
 - Modélisation des architectures cibles
 - Mesures directes

Estimation du temps d'exécution d'une application

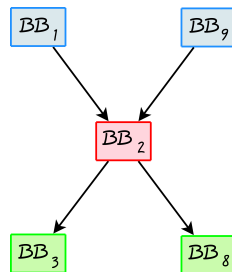
Estimation du WCET d'une application

- Exemple d'analyse statique du WCET

- 3. Calcul du WCET → Implicit Path Enumeration Technique

- Contraintes structurelles :

$$\left\{ \begin{array}{lcl} N_1^{BB} + N_9^{BB} & = & N_2^{BB} \\ N_2^{BB} & = & N_3^{BB} + N_8^{BB} \\ N_3^{BB} + N_{10}^{BB} & = & N_4^{BB} \\ N_4^{BB} & = & N_5^{BB} + N_9^{BB} \\ N_5^{BB} + N_7^{BB} & = & N_6^{BB} \\ N_6^{BB} & = & N_7^{BB} + N_{10}^{BB} \end{array} \right.$$



Estimation du temps d'exécution d'une application

Estimation du WCET d'une application

- Exemple d'analyse statique du WCET
 - 3. Calcul du WCET → Implicit Path Enumeration Technique
 - Contraintes fonctionnelles :

$$\left\{ \begin{array}{lcl} N_1^{BB} & = & 1 \\ N_3^{BB} & = & \text{dimensions} \\ N_3^{BB} & = & \text{dimensions}^2 \\ N_3^{BB} & = & \text{dimensions}^3 \end{array} \right.$$

Estimation du temps d'exécution d'une application

Estimation du WCET d'une application

- Exemple d'analyse statique du WCET
 - **3. Calcul du WCET** → **Implicit Path Enumeration Technique**
 - Problème d'optimisation linéaire à variables entières sous contraintes :

$$WCET = \sum_b N_b^{BB} \cdot T_b^{BB}$$

Estimation du temps d'exécution d'une application

Prédiction basée sur un historique

Principe :

Deux exécutions d'un programme dans des contextes proches produisent des temps d'exécution voisins

Estimation du temps d'exécution d'une application

Prédiction basée sur un historique

- Approche par apprentissage basé sur des instances
 - Base de connaissances contenant les expériences passées
 - Prédiction réalisée à partir des expériences les plus appropriées

Méthode hybride de prédiction

Méthode hybride de prédiction

- Approche basée sur :
 - le profil des applications
 - un historique d'exécutions passées
- Etapes nécessaires :
 - 1 Découpage du programme en bloc de base

$$T_{App}(E) = \sum_{b \in \mathbb{B}} N_b^{BB}(E) \cdot T_b^{BB}$$

- 2 Détermination du temps d'exécution des blocs de base
- 3 Détermination du nombre d'exécutions des blocs de base
- 4 Estimation du temps d'exécution global de l'application

Plan de la présentation

- 1 Estimation du temps d'exécution d'une application
- 2 **Résolution du modèle de prédiction hybride**
 - Détermination du temps d'exécution des blocs de base
 - Etude du comportement d'un programme
 - Modèle complet de prédiction hybride
- 3 Gestion des ressources d'une grille à l'aide d'un modèle économique
- 4 Mise en place d'un gestionnaire de ressources
- 5 Conclusion et perspectives

Introduction

Modèle de prédiction hybride de temps d'exécution

$$T_{App}(E) = \sum_{b \in \mathbb{BB}} N_b^{BB}(E) \cdot T_b^{BB}$$

Hypothèses

- $E \simeq E^* \implies T_{App}(E) \simeq T_{App}(E^*)$
- $E \simeq E^* \implies N_b^{BB}(E) \simeq N_b^{BB}(E^*)$
- T_b^{BB} constant

Détermination du temps d'exécution des blocs de base

Obtention d'un système d'équations linéaires

- X exécutions du programme pour différents jeux d'entrées $E^{(e)}$

$$\forall e \in [1, X] \quad T_{App}(E^{(e)}) = \sum_{b \in \mathbb{B}} N_b^{BB}(E^{(e)}) \cdot T_b^{BB}$$

- Utilisation d'outils de *profiling* :
 - $G_{prof} : T_{App}(E^{(e)})$
 - $G_{cov} : N_b^{BB}(E^{(e)})$

Détermination du temps d'exécution des blocs de base

Obtention d'un système d'équations linéaires

- Système d'équations linéaires : $A \cdot x = b$

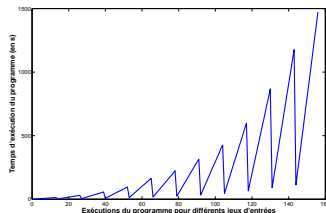
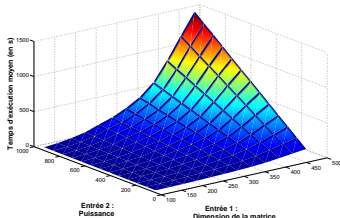
$$A = \begin{bmatrix} N_{b_1}^{BB}(E^{(1)}) & N_{b_2}^{BB}(E^{(1)}) & \dots & N_{b_{Card(\mathbb{B})}}^{BB}(E^{(1)}) \\ N_{b_1}^{BB}(E^{(2)}) & N_{b_2}^{BB}(E^{(2)}) & \dots & N_{b_{Card(\mathbb{B})}}^{BB}(E^{(2)}) \\ \vdots & \vdots & \vdots & \vdots \\ N_{b_1}^{BB}(E^{(X)}) & N_{b_2}^{BB}(E^{(X)}) & \dots & N_{b_{Card(\mathbb{B})}}^{BB}(E^{(X)}) \end{bmatrix}$$

$$b = \begin{bmatrix} T_{App}(E^{(1)}) \\ T_{App}(E^{(2)}) \\ \vdots \\ T_{App}(E^{(X)}) \end{bmatrix} \quad x = \begin{bmatrix} T_{b_1}^{BB} \\ T_{b_2}^{BB} \\ \vdots \\ T_{b_{Card(\mathbb{B})}}^{BB} \end{bmatrix}$$

Détermination du temps d'exécution des blocs de base

Exemple traité

- Programme d'élévation d'une matrice carrée de dimension quelconque à une puissance donnée
- Deux entrées :
 - dimension de la matrice
 - puissance à laquelle l'élever



Détermination du temps d'exécution des blocs de base

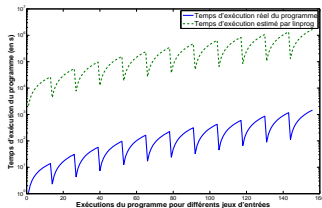
Résolution à l'aide d'outils standards

- Outils utilisés :
 - `linprog` : solveur de problèmes d'optimisation ou de programmation linéaire
 - `fsolve` : solveur d'équations non-linéaires
- Méthode :
 - 1 Résoudre le système $A \cdot x = b$
 - 2 Comparer le produit $A \cdot x$ avec b

Détermination du temps d'exécution des blocs de base

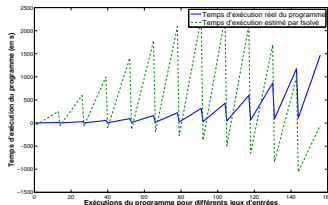
Résolution à l'aide d'outils standards

- Fonction linprog



Erreur relative moyenne : $1,50 \times 10^5\%$

- Fonction fsolve



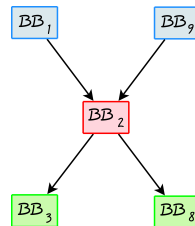
Erreur relative moyenne : $7,06 \times 10^2\%$

Détermination du temps d'exécution des blocs de base

Résolution à l'aide d'outils standards

- Causes des erreurs observées :
 - Système mal conditionné
 - Dépendances entre lignes et colonnes dûes :
 - aux jeux d'entrées choisis
 - à la structure du programme

$$\left\{ \begin{array}{l} N_1^{BB} + N_9^{BB} = N_2^{BB} \\ N_2^{BB} = N_3^{BB} + N_8^{BB} \end{array} \right.$$



Détermination du temps d'exécution des blocs de base

Résolution itérative du système d'équations

- Système à résoudre :

$$A \cdot x = b$$

Objectif

Exprimer le système d'équations sous la forme d'une suite convergente :

$$\begin{cases} x_0 : \textit{conditions initiales} \\ x_i = G \cdot x_{i-1} + h \end{cases}$$

Détermination du temps d'exécution des blocs de base

Résolution itérative du système d'équations

- ④ Décomposition de A en valeurs singulières

$$A = U \cdot \Sigma \cdot V^T$$

Détermination du temps d'exécution des blocs de base

Résolution itérative du système d'équations

- 1 Décomposition de A en valeurs singulières
- 2 Introduction d'une erreur ε

$$A = U \cdot (\Sigma + \varepsilon) \cdot V^T$$

Détermination du temps d'exécution des blocs de base

Résolution itérative du système d'équations

- 1 Décomposition de A en valeurs singulières
- 2 Introduction d'une erreur ε
- 3 Décomposition de Σ

$$\Sigma = \alpha \cdot I_{m \times n} - D$$

Détermination du temps d'exécution des blocs de base

Résolution itérative du système d'équations

- 1 Décomposition de A en valeurs singulières
- 2 Introduction d'une erreur ε
- 3 Décomposition de Σ

$$x = \alpha^{-1} \cdot V \cdot I_{n \times m} \cdot (D - \varepsilon) \cdot V^T \cdot x + \alpha^{-1} \cdot V \cdot I_{n \times m} \cdot U^T \cdot b$$

Détermination du temps d'exécution des blocs de base

Résolution itérative du système d'équations

$$\begin{cases} x_0 : \text{conditions initiales} \\ x_i = G \cdot x_{i-1} + h \end{cases}$$

- Avec :

- $G = \alpha^{-1} \cdot V \cdot I_{n \times m} \cdot (D - \varepsilon) \cdot V^T$

- $h = \alpha^{-1} \cdot V \cdot I_{n \times m} \cdot U^T \cdot b$

Détermination du temps d'exécution des blocs de base

Résolution itérative du système d'équations

$$G = \alpha^{-1} \cdot V \cdot I_{n \times m} \cdot (D - \varepsilon) \cdot V^T$$

Théorème

La suite converge si le rayon spectral de la matrice G , noté $\rho(G)$, est strictement inférieur à 1.

Propriété

La vitesse de convergence de la suite est d'autant plus élevée que le rayon spectral de la matrice G est faible.

Détermination du temps d'exécution des blocs de base

Résolution itérative du système d'équations

$$G = \alpha^{-1} \cdot V \cdot I_{n \times m} \cdot (D - \varepsilon) \cdot V^T$$

Proposition

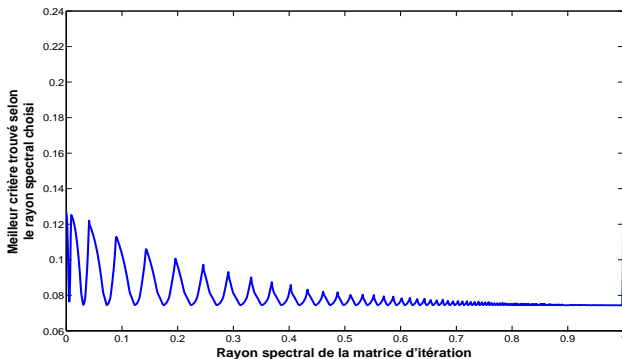
Etant donnée une valeur $\rho(G)$ du rayon spectral de la matrice G , une valeur possible de la matrice d'erreur ε pour l'obtenir est :

$$\varepsilon = (1 - \rho(G)) \cdot D$$

Détermination du temps d'exécution des blocs de base

Résolution itérative du système d'équations

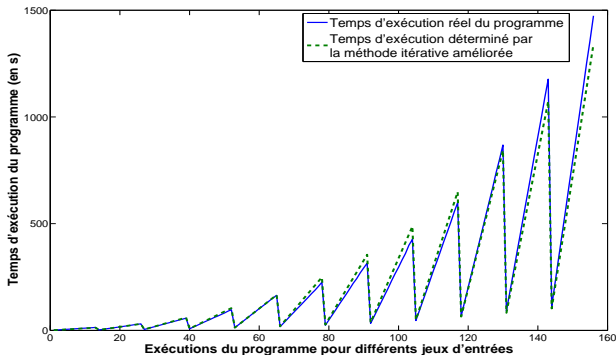
- Convergence de la suite :



Détermination du temps d'exécution des blocs de base

Résolution itérative du système d'équations

- Résultats obtenus :



Détermination du temps d'exécution des blocs de base

Etude du comportement d'un programme

$$T_{App}(E) = \sum_{b \in \mathbb{B}} N_b^{BB}(E) \cdot T_b^{BB}$$

Estimation du chemin d'exécution emprunté en fonction des entrées du programme

- Approche d'apprentissage basé sur des instances

Etude du comportement d'un programme

Modèle mathématique

- Vecteur d'entrées :

$$E^{(i)} = \left\{ E_v^{(i)} \right\}_{v \in [1; N^V]} = \begin{bmatrix} E_1^{(i)} \\ E_2^{(i)} \\ \vdots \\ E_{N^V}^{(i)} \end{bmatrix}$$

- Base de connaissances :

$$\mathbb{E}^{Exp} = \bigcup_{i \in [1; X]} \left\{ E^{(i)} \right\}$$

Etude du comportement d'un programme

Modèle mathématique

- Estimation du nombre d'exécutions des blocs de base :

$$N_b^{BB}(E^*) = \frac{\sum_{E \in \mathbb{R}^{Exp}} \left[K(D_b(E^*, E)) \cdot N_b^{BB}(E) \right]}{\sum_{E \in \mathbb{R}^{Exp}} K(D_b(E^*, E))}$$

- Fonction de pondération :

$$K(d) = e^{-\left(\frac{d}{k}\right)^2}$$

Etude du comportement d'un programme

Modèle mathématique

- Distance entre deux entrées :

$$D_b(E^{(1)}, E^{(2)}) = \sqrt{\sum_{v=1}^{N^V} w_{v,b} \cdot d(E_v^{(1)}, E_v^{(2)})^2} \quad \text{avec} \quad 0 \leq w_{v,b} \leq 1$$

- Métrique hétérogène de distance :

- $0 \leq d(E_v^{(1)}, E_v^{(2)}) \leq 1$
- Valeurs numériques : $\frac{|E_v^{(1)} - E_v^{(2)}|}{\max_{E_v^{(x)}} - \min_{E_v^{(x)}}}$
- Valeurs nominales :
 - $E_v^{(1)} = E_v^{(2)} \Rightarrow d(E_v^{(1)}, E_v^{(2)}) = 0$
 - $E_v^{(1)} \neq E_v^{(2)} \Rightarrow d(E_v^{(1)}, E_v^{(2)}) = 1$

Etude du comportement d'un programme

Modèle mathématique

$$D_b(E^{(1)}, E^{(2)}) = \sqrt{\sum_{v=1}^{N^V} w_{v,b} \cdot d(E_v^{(1)}, E_v^{(2)})^2} \quad \text{avec } 0 \leq w_{v,b} \leq 1$$

```
for (int i = 0 ; i < uneEntree ; i++) { // avec 0 ≤ uneEntree ≤ 100
    instructions ;
}
for (int j = 0 ; j < autreEntree ; j++) { // avec 0 ≤ autreEntree ≤ 100
    instructions ;
}
```

$$E_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad E_2 = \begin{bmatrix} 50 \\ 100 \end{bmatrix} \quad E^* = \begin{bmatrix} 0 \\ 100 \end{bmatrix}$$

Etude du comportement d'un programme

Annotations du code source

Déterminer l'impact relatif des entrées sur le nombre d'exécutions des blocs de base

↪ Attribuer une valeur à $w_{v,b}$

- Spécifications :
 - Permettre à l'utilisateur d'exprimer les dépendances des blocs de base vis-à-vis des entrées
 - Exprimer les dépendances directement dans le code source
 - Système transparent vis-à-vis de la compilation et de l'exécution de l'application

Etude du comportement d'un programme

Annotations du code source

- Exemples d'annotations :

```
#pragma etp inputs (uneEntree, autreEntree)
```

```
#pragma etp begin dependency (uneEntree)
for (int i = 0; i < uneEntree; i++) {
    instructions;
}
#pragma etp end
```

Etude du comportement d'un programme

Exemple traité

- Application de filtrage particulière
 - Application parallèle adaptée à une exécution sur grille
 - Modèle maître / esclave
 - Tâches esclaves identiques et peu communicantes
 - Temps d'exécution de la tâche maître négligeable devant celui des tâches esclaves

Application parallèle à gros grain dont les tâches sont identiques et synchrones

Etude du comportement d'un programme

Exemple traité

- Application de filtrage particulière
- Entrées du programme :
 - Nombre de particules
 - Nombre d'intervalles de temps
 - Nombre de tâches

$$\mathbb{E} = \left\{ E = \begin{bmatrix} E_1 \\ E_2 \\ E_3 \end{bmatrix} \text{ tels que } \begin{array}{ccccc} 50000 & \leq & E_1 & \leq & 500000 \\ 1000 & \leq & E_2 & \leq & 10000 \\ 4 & \leq & E_3 & \leq & 12 \end{array} \right\}$$

- Programme correctement annoté

Etude du comportement d'un programme

Résultats obtenus

- Contexte :
 - $X = 250$
 - $k = 0,01$
 - 500 requêtes

- Résultats :

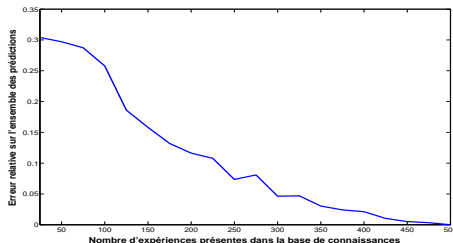
Nombre de requêtes	Erreurs
300	< 1%
40	entre 1% et 5%
43	entre 5% et 10%
67	entre 10% et 25%
50	> 25%

- Erreur moyenne : 6,61 %

Etude du comportement d'un programme

Résultats obtenus

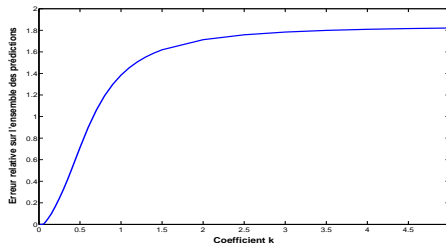
- Influence du contenu de la base de connaissances
 - Erreur en fonction du nombre d'expériences contenues dans la base de connaissances



Etude du comportement d'un programme

Résultats obtenus

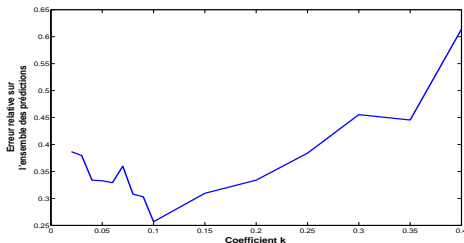
- Influence de la largeur de la fonction de pondération
 - Erreur en fonction de la valeur de k avec $X = 500$



Etude du comportement d'un programme

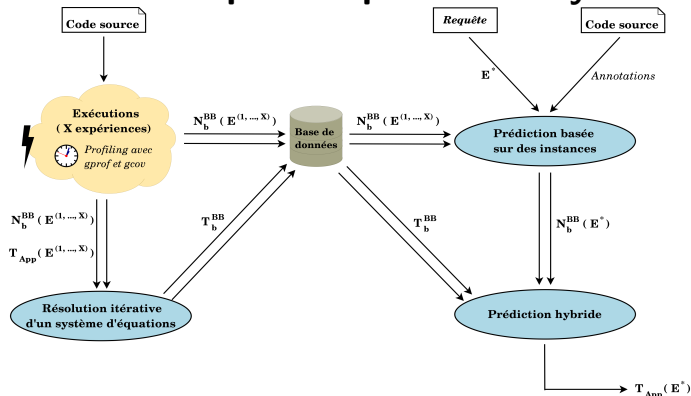
Résultats obtenus

- Influence de la largeur de la fonction de pondération
 - Erreur en fonction de la valeur de k avec $X = 75$



Modèle complet de prédiction hybride

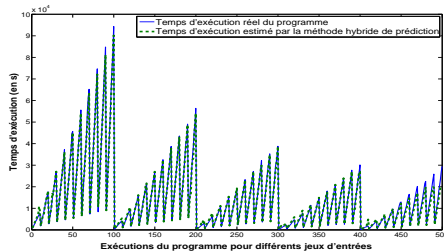
Modèle complet de prédiction hybride



Modèle complet de prédiction hybride

Résultats obtenus

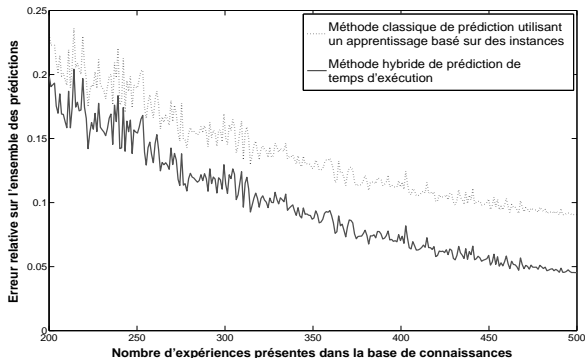
- Caractéristiques des prédictions effectuées
 - Nombre d'expériences contenues dans la base : 350
 - Nombre de requêtes : 250



- Résultats :
 - Erreur moyenne : 8,70 %

Modèle complet de prédiction hybride

Résultats obtenus



Plan de la présentation

- 1 Estimation du temps d'exécution d'une application
- 2 Résolution du modèle de prédiction hybride
- 3 Gestion des ressources d'une grille à l'aide d'un modèle économique
 - La problématique de l'ordonnancement
 - Introduction aux modèles économiques
 - Proposition d'un modèle économique
- 4 Mise en place d'un gestionnaire de ressources
- 5 Conclusion et perspectives

Gestion des ressources d'une grille à l'aide d'un modèle économique

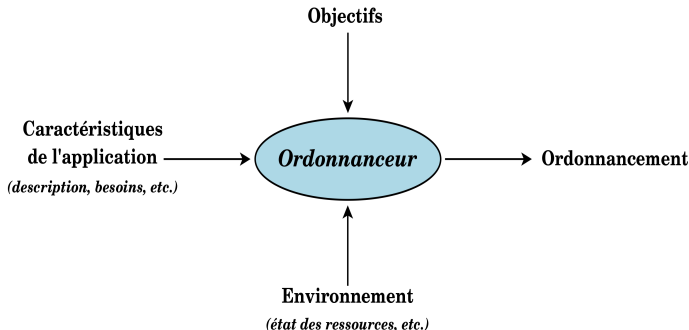
Objectif

Mettre en place un **modèle d'ordonnancement** basé sur des **paradigmes économiques**

↪ *Réguler la demande sur les ressources*

La problématique de l'ordonnancement

Principe général



La problématique de l'ordonnancement

Stratégies d'ordonnancement

- Politique FIFO
- Backfilling
- Politique MET (*Minimum Execution Time*)
- Politique MCT (*Minimum Completion Time*)

La problématique de l'ordonnancement

Stratégies d'ordonnancement

- Politique FIFO
- Backfilling
- Politique MET (*Minimum Execution Time*)
- Politique MCT (*Minimum Completion Time*)

Stratégies difficiles à mettre en œuvre dans des environnements de grilles

Introduction aux modèles économiques

Objectifs principaux des modèles économiques

- Mettre en relation producteurs et consommateurs de ressources
- Réguler l'offre et la demande
- S'affranchir d'un contrôle centralisé
- Gérer l'hétérogénéité des ressources

Introduction aux modèles économiques

Modèles fondamentaux

- Modèle de marché
- Modèle des négociations
- Modèle de l'appel d'offres
- Modèle de la vente aux enchères

Proposition d'un modèle économique

Contexte

- Ressources
 - Ressources hétérogènes
 - Fonctionnement en mode partagé
 - Grille dédiée \Rightarrow Modèle de prédiction de charge
- Applications
 - Applications parallèles à gros grain
 - Tâches identiques et synchrones
- Modèle économique
 - Modèle de marché
 - Prix des ressources variable au cours du temps
 - Prise en compte de l'état futur des ressources
 - Détermination d'un placement et d'une date de début
 - Minimisation d'un compromis temps de réponse / coût

Proposition d'un modèle économique

Principe de fonctionnement

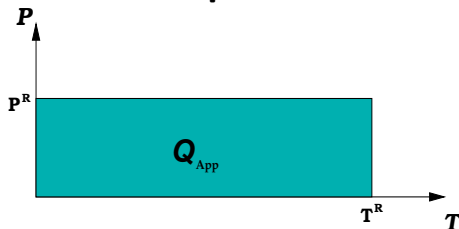
Exprimer le temps de réponse et le coût de l'application en fonction de l'ordonnancement choisi

Problème d'optimisation

- Trouver l'ordonnancement qui minimise un compromis entre temps de réponse et coût

Proposition d'un modèle économique

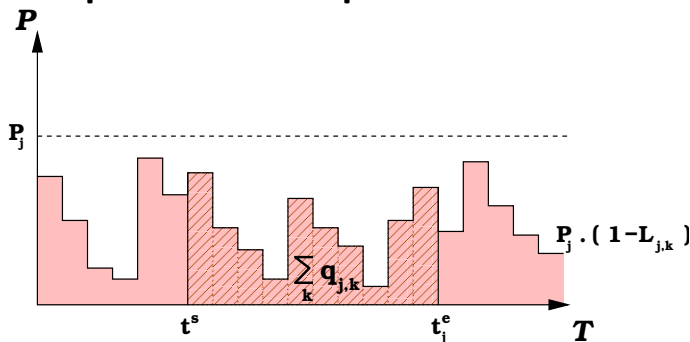
Notion de quantité de calculs



$$Q_{App} = T^R \cdot P^R \quad \Rightarrow \quad Q = \frac{T^R \cdot P^R}{A}$$

Proposition d'un modèle économique

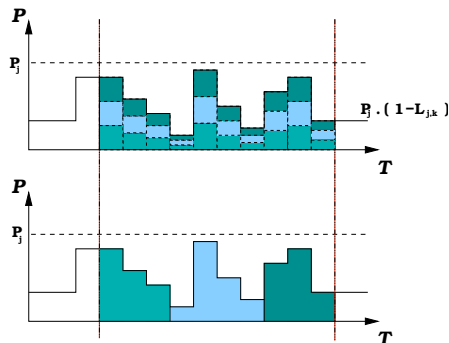
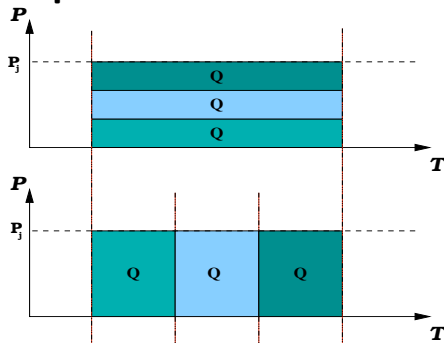
Temps nécessaires pour exécuter une tâche



$$\min \left\{ K \quad / \quad \sum_{k=k^S}^{k^S+K-1} \Delta T \cdot P_j \cdot (1 - L_{j,k}) \geq \frac{T^R \cdot P^R}{A} \right\}$$

Proposition d'un modèle économique

Exécution de plusieurs tâches sur un même processeur



Proposition d'un modèle économique

Temps de réponse de l'application

$$T_{App} = t^S + T^E + T^C$$

où :

- t^S = date de début
- T^E = temps d'exécution
- T^C = temps de communication

Proposition d'un modèle économique

Coût de l'application

- Coût lié au temps processeur consommé
- Coût lié au nombre de processeurs utilisés
- Coût lié à la puissance des processeurs utilisés

$$C^T \cdot t^{CPU} \frac{T^R \cdot P^R}{A} \cdot \sum_{j=1}^{N_A^P} \frac{u_j}{P_j} + C^N \cdot N_U^P + C^P \cdot \frac{t_{exec}^E}{t_{exec}^{EC}} \cdot \sum_{j=1}^{N_A^P} \left[u_j \cdot P_j \cdot \left(\frac{1}{t_{exec}^{EC}} \cdot \sum_{k=k^S}^{k^{EC}} C_{j,k}^P \cdot \Delta T \right) \right]$$

Proposition d'un modèle économique

Problème d'optimisation :

- non-linéaire
- sous contraintes
- à variables entières

Plan de la présentation

- 1 Estimation du temps d'exécution d'une application
- 2 Résolution du modèle de prédiction hybride
- 3 Gestion des ressources d'une grille à l'aide d'un modèle économique
- 4 Mise en place d'un gestionnaire de ressources
 - Implémentation du modèle économique
 - Intégration au sein d'OAR
 - Résultats obtenus
- 5 Conclusion et perspectives

Objectifs et contexte

Missions d'un gestionnaire de ressources

- Service d'ordonnancement
- Service d'information
- Service d'exécution et d'observation

Objectifs et contexte

Objectifs

- 1 Implémentation du modèle économique
- 2 Intégration au sein d'OAR

Implémentation du modèle économique

Problème d'optimisation :

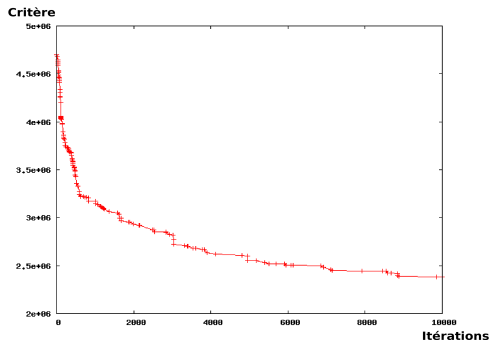
- non-linéaire
 - sous contraintes
 - à variables entières
-
- Espace de recherche important
 - Rapidité d'exécution / Optimalité des solutions trouvées
 - Adaptation aux structures de données du modèle

Utilisation d'un **algorithme génétique**

Implémentation du modèle économique

Performances de l'algorithme

- Convergence de l'algorithme



- Application de 40 tâches
- Grille de 1000 processeurs
- Résultats corrects vers 5000 itérations
- Stabilité autour de 9000 itérations

Implémentation du modèle économique

Performances de l'algorithme

Hôte d'exécution : *Intel Dual-Core 3,2 GHz*

- Application de 40 tâches
- Grille de 1000 processeurs
- **Temps de résolution : 5 secondes**

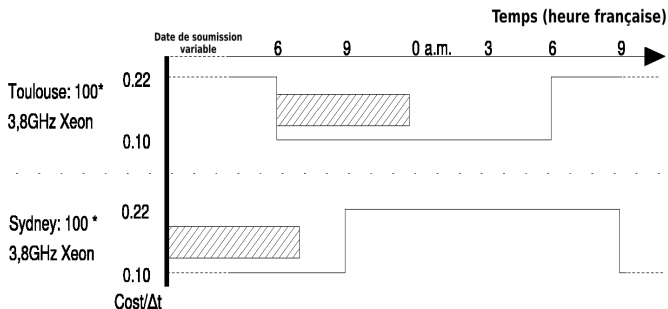
- Application de 40 tâches
- Grille de 5000 processeurs
- **Temps de résolution : 18 secondes**

Intégration au sein d'OAR

Spécifications de l'intégration du modèle au sein d'OAR

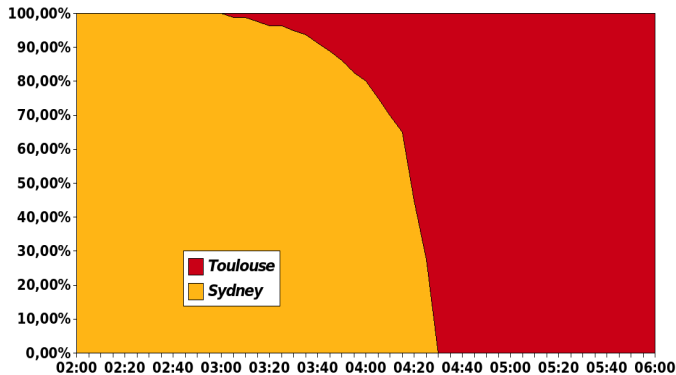
- Soumission d'applications via OAR
 - Calcul du placement et la date de début
 - Utilisation du modèle économique (algorithme génétique)
- Modifications d'OAR
 - Adaptations au modèle économique
 - Mode partagé
 - Prédiction des performances des ressources
 - Modifications non-intrusives
 - Découpler au maximum le cycle de vie d'OAR de celui de notre ordonnanceur

Résultats obtenus



Expérience : $W^C = 0,9$; $W^T = 0,1$

Résultats obtenus



Plan de la présentation

- 1 Estimation du temps d'exécution d'une application
- 2 Résolution du modèle de prédiction hybride
- 3 Gestion des ressources d'une grille à l'aide d'un modèle économique
- 4 Mise en place d'un gestionnaire de ressources
- 5 Conclusion et perspectives
 - Conclusion
 - Perspectives

Conclusion

Placement à l'aide de modèles économiques

Définition d'un modèle d'ordonnancement

- **Modèle économique de marché**, prix des ressources variable

Calcul d'ordonnancements

- Minimisation d'un compromis **temps de réponse / coût**
- Choix des **ressources** à utiliser
- Choix d'une **date de début**

Implémentation

- Algorithme génétique
- Intégration au sein d'OAR

Conclusion

Placement à l'aide de modèles économiques

Apports

- **Gestionnaire de ressources** complet et fonctionnel
- Ordonnancements à partir de **critères de performances et de coût**
- Prise en compte de l'**évolution future des ressources** (prix et charge)
- Possibilité de **différer l'exécution** d'une application

Conclusion

Prédiction de comportement d'applications parallèles

Méthode hybride de prédiction de temps d'exécution

- Approche utilisant un **historique d'exécutions passées** (apprentissage basé sur des instances)
- Approche basée sur le **profil des applications**

Conclusion

Prédiction de comportement d'applications parallèles

Apports

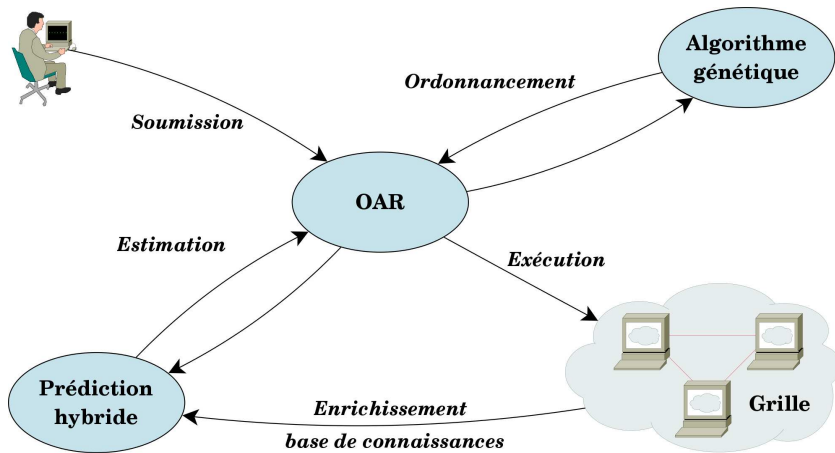
- **Prédiction du temps d'exécution** d'une application en fonction de ses entrées
- Estimation basée sur un **historique d'exécutions passées**
- **Prise en compte de la structure du programme**
- **Précision accrue**

Conclusion

Proposition de modèles permettant de **réguler la demande sur les ressources d'une grille**

- **Facturation** des ressources utilisées
- **Estimation** des besoins en ressources

Conclusion



Perspectives

Perspectives

Modèle économique

- Extension du type de ressources considérées
- Augmentation du coût en fonction de la charge des ressources
- Prise en compte de nouveaux types d'applications

Prédiction du temps d'exécution

- Instrumentation du code objet
- Annotations automatiques
- Modèles analytiques

Questions...