

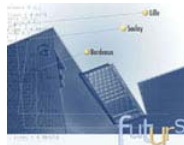


Résolution exacte de problèmes d'optimisation combinatoire sur grilles de calcul

M. MEZMAZ, N. MELAB et E-G. TALBI

{mezmaz,melab,talbi}@lifl.fr

Lille, 30 octobre 2006



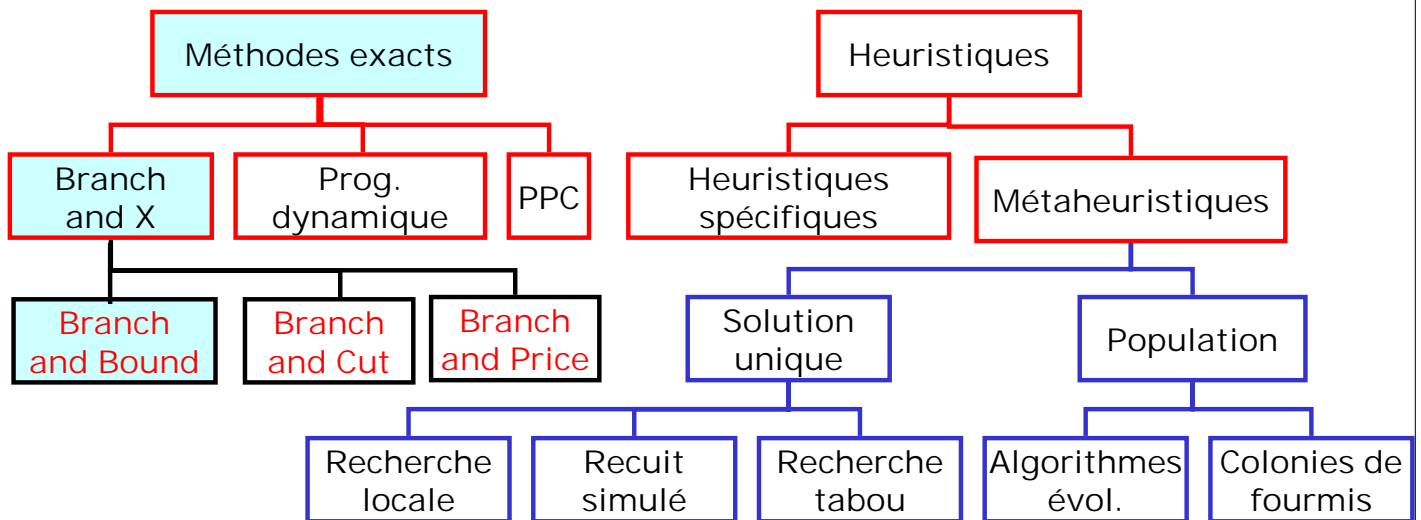
Motivations

- Problèmes d'optimisation complexes et de grande taille dans de nombreux secteurs de l'industrie
→ Télécommunications, bioinformatique, transport, ...
- Traitement efficace et flexible d'une combinatoire importante (Parallélisme à grande échelle)

Problème mono-objectif (POC) $\min f(x) \quad x \in S$

Problème multi-objectif (PMO) $\left\{ \begin{array}{l} \min f(x) = (f_1(x), f_2(x), \dots, f_n(x)) \quad n \geq 2 \\ \text{s.c. } x \in S \end{array} \right.$

Une taxonomie des méthodes de résolution



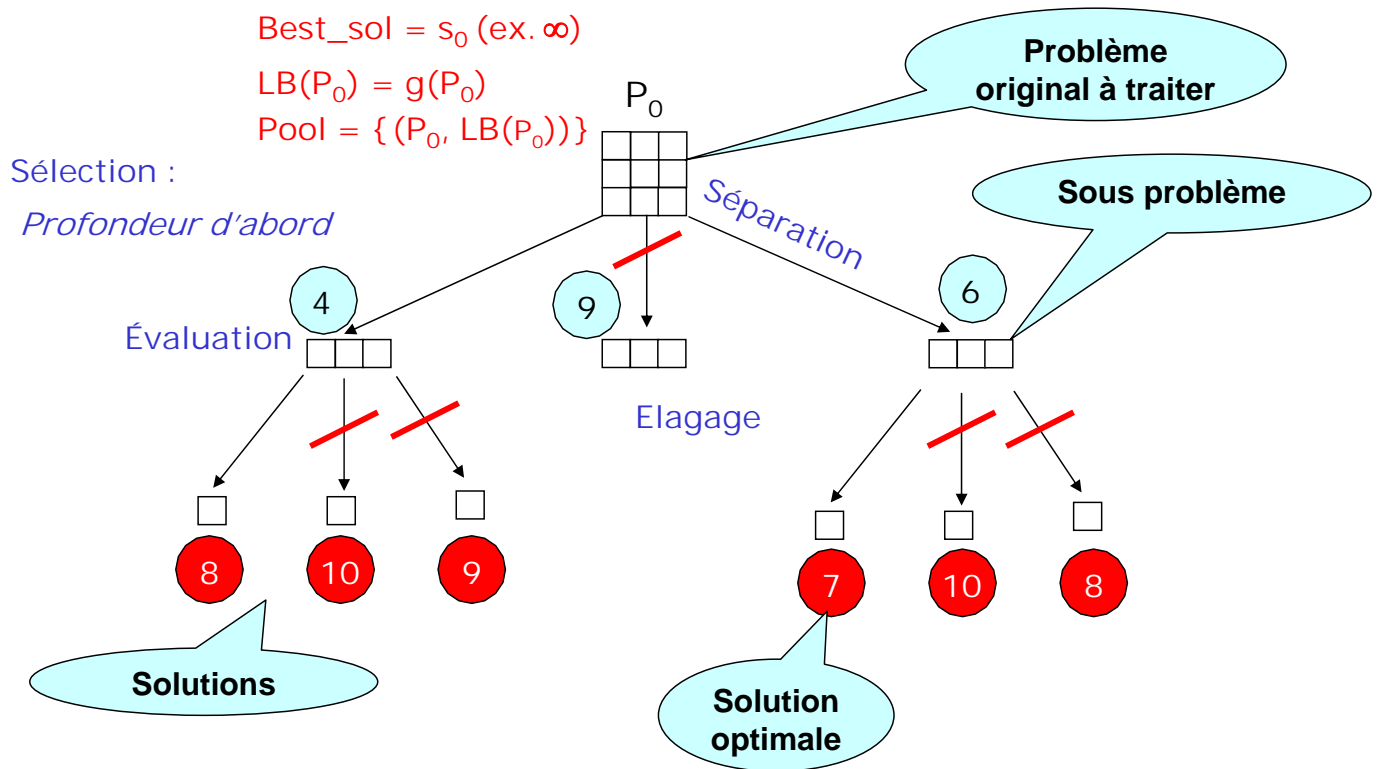
Heuristiques : solutions approchées sur des instances de grande taille

Méthodes exactes : optimalité et exploitation sur des instances de taille plus raisonnable

Branch-and-Bound (B&B) - Opérateurs

- Séparation ou décomposition (*Branching*)
 - Génération de sous-problèmes indépendants (nœuds fils) par ajout de contraintes à un problème (nœud père)
- Evaluation (*Bounding*)
 - Calcul de la borne inférieure $LB(P_i)$
- Elagage
 - P_i élagué si $LB(P_i) \geq Best_sol$
- Sélection
 - Quel sous-problème choisir parmi une liste de sous-problèmes à traiter ?
 - Stratégies : {profondeur, meilleur, largeur} d'abord, ...

Branch-and-Bound (B&B) - Illustration



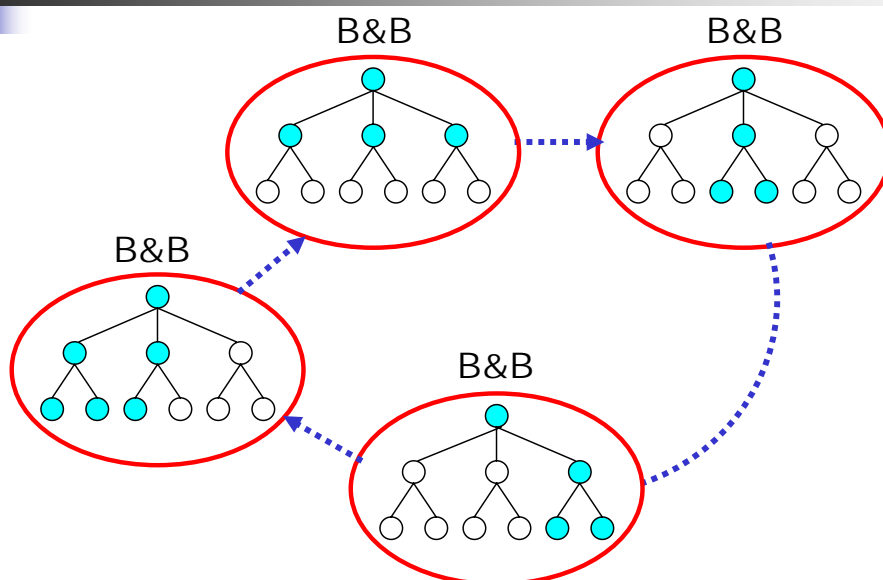
Plan

- Modèles parallèles pour les méthodes exactes
- Gridification de l'algorithme B&B
- Application au problème du Flow-Shop
- Conclusions et perspectives

Modèles parallèles pour les méthodes exactes

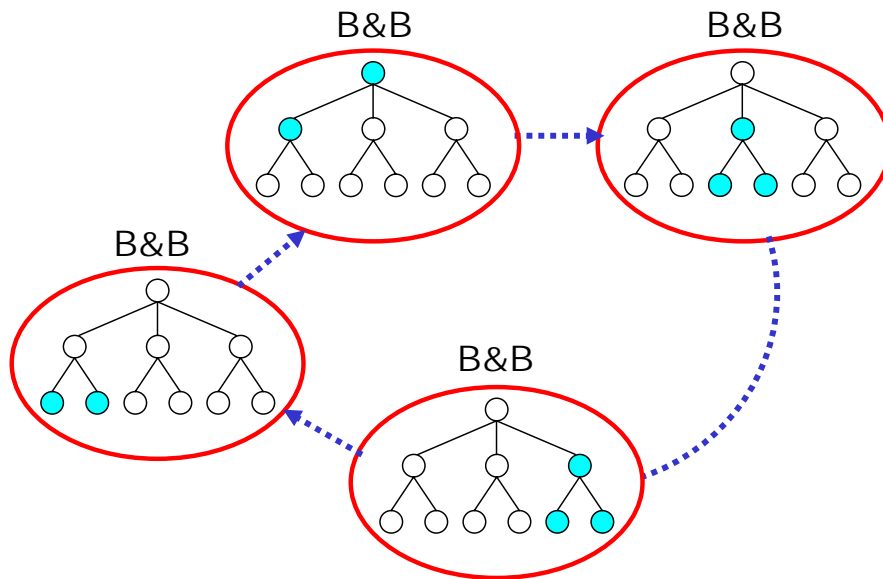
- Analyse inspirée des algorithmes B&B
- Modèles parallèles
 - Modèle multi-paramétrique parallèle
 - Exploration parallèle arborescente
 - Évaluation parallèle des bornes
 - Identique au modèle d'évaluation parallèle d'une population des métaheuristiques
 - Évaluation parallèle d'une borne
 - Identique au modèle d'évaluation parallèle d'une solution des métaheuristiques

Modèle multi-paramétrique parallèle



- Plusieurs B&B indépendants ou coopératifs (e.g. avec stratégies de sélection – profondeur d'abord, meilleur d'abord, ...)
- Conception réutilisable (modèle générique)
- Exploration redondante
- Exploitable sur grille s'il est combiné avec d'autres modèles

Exploration parallèle arborescente



- Le modèle le plus utilisé et étudié
- Parallélisme massif nécessitant une grille de calcul
- Justifie à lui tout seul l'intérêt d'une grille de calcul
- Mode synchrone inefficace (*Gendron et al. 94*)

Caractéristiques des grilles

- Multi-domaine d'administration
 - Sécurité (passage de pare-feu), différents systèmes d'ordonnancement, ...
- Grande échelle
 - Délais de communication importants, ...
- Volatile (disponibilité variable et pannes des machines)
 - Découverte de ressources, checkpointing, re-ordonnancement, ...
- Hétérogène (systèmes, réseaux, machines, etc.)
 - Equilibrage de charge, la mesure des performances, ...



Exploration parallèle sur grille - Problématique

- Régulation dynamique de charge
 - **Arbre irrégulier** dans un contexte hétérogène volatile à grande échelle
- Tolérance aux pannes
 - **Recherche des solutions exacte** dans un contexte volatile
- Partage des solutions ...
 - ... extensibilité dans un **contexte à grande échelle**
- Détection de la terminaison
 - **Modèle asynchrone**



Plan

- **Modèles parallèles pour les méthodes exactes**
- **Gridification de l'algorithme B&B**
- **Application au problème du Flow-Shop**
- **Conclusions et perspectives**



Approche proposée - Objectif

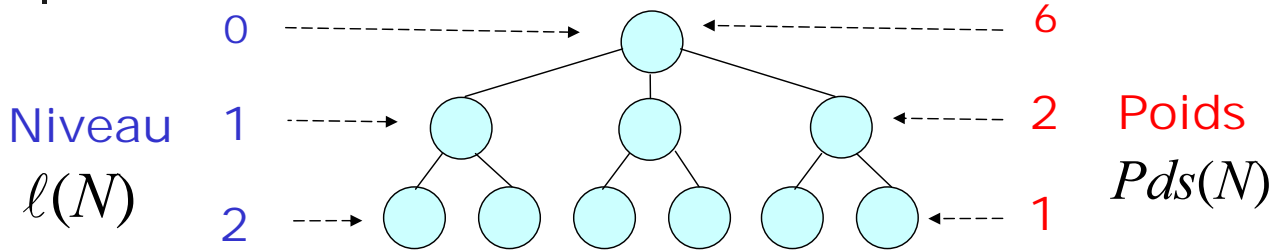
- Optimisation la de distribution du travail
 - Communications (transferts), hétérogénéité (puissance de calcul)
- Proposition d'un mécanisme de *sauvegarde/restauration* à faible coût (stockage et communications)
- Partage efficace de la *meilleure solution trouvée*
- Détection implicite et naturelle de la terminaison



Approche proposée – Principe

- Caractéristiques
 - Approche *Dispatcher-Worker*
 - Exploration de l'*arbre de base* en *profondeur d'abord*
- Basée sur un **codage** particulier ...
 - ... des noeuds
 - Chaque nœud porte un numéro
 - ... des sous-arbres de l'arbre de base
 - un *intervalle* est associé à chaque nœud N
 - Cet intervalle décrit le sous-arbre dont N est la racine

Approche proposée – Codage [Poids]



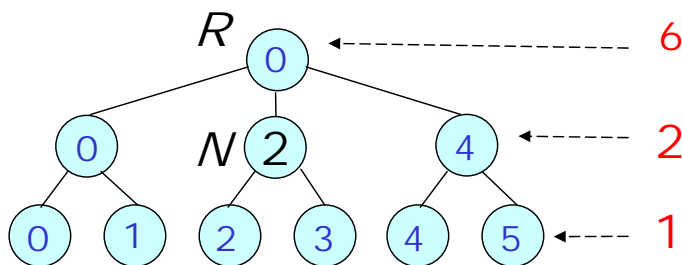
- Cas général

$$Pds(N) = \begin{cases} 1 & \text{si } \mathbf{Fils}(N) = \phi \\ \sum_{f \in \mathbf{Fils}(N)} Pds(f) & \text{sinon} \end{cases}$$

- Cas des problèmes de permutation de taille T

$$Pds(N) = (T - \ell(N))! \\ \text{avec } \ell(N) = 0, 1, \dots$$

Approche proposée – Codage [Numéro]



- Numéro d'un nœud

$$\text{Num}(N) = \sum_{n \in \mathbf{Chemin}(N)} \text{Rg}(n) * Pds(n)$$

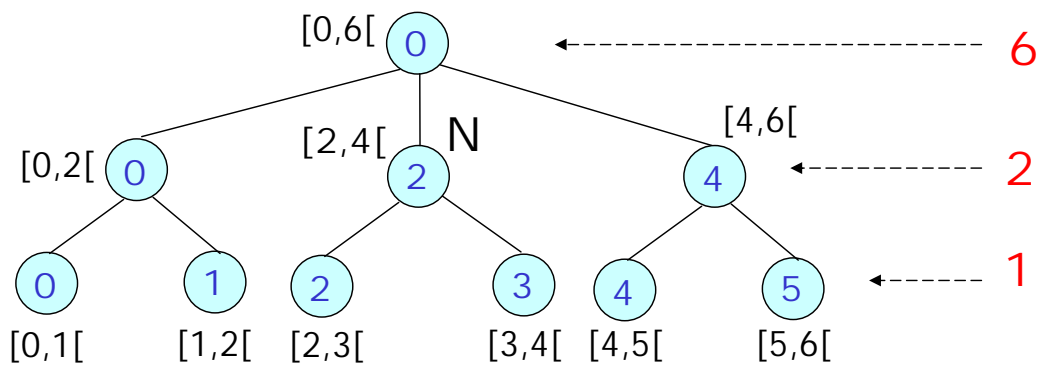
- $\mathbf{Chemin}(N)$: l'ensemble des nœuds rencontrés en partant de la racine R au nœud N .
- $\text{Rg}(N)$: sa position parmi ses nœuds frères

- Exemple

$$0 \leq \text{Rg}(N) \leq |\{n / \ell(n) = \ell(N)\}| - 1$$

$$\text{Num}(N) = 0 * 6 + 1 * 2 = 2$$

Approche proposée – Codage [Intervalle]



- Intervalle d'un noeud

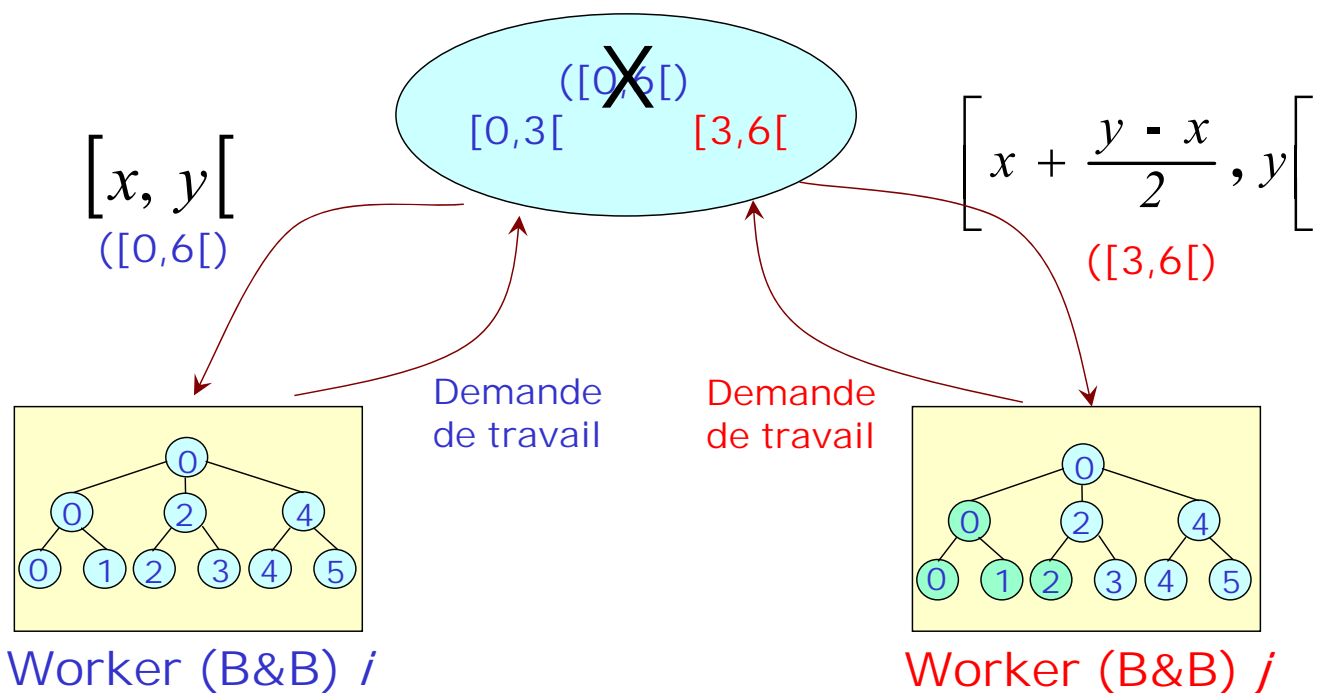
$$\text{Int}(N) = [\text{Num}(N), \text{Num}(N) + \text{Pds}(N)[$$

- Exemple

$$\text{Int}(N) = [2, 2 + 2[= [2, 4[$$

Répartition de charge (1)

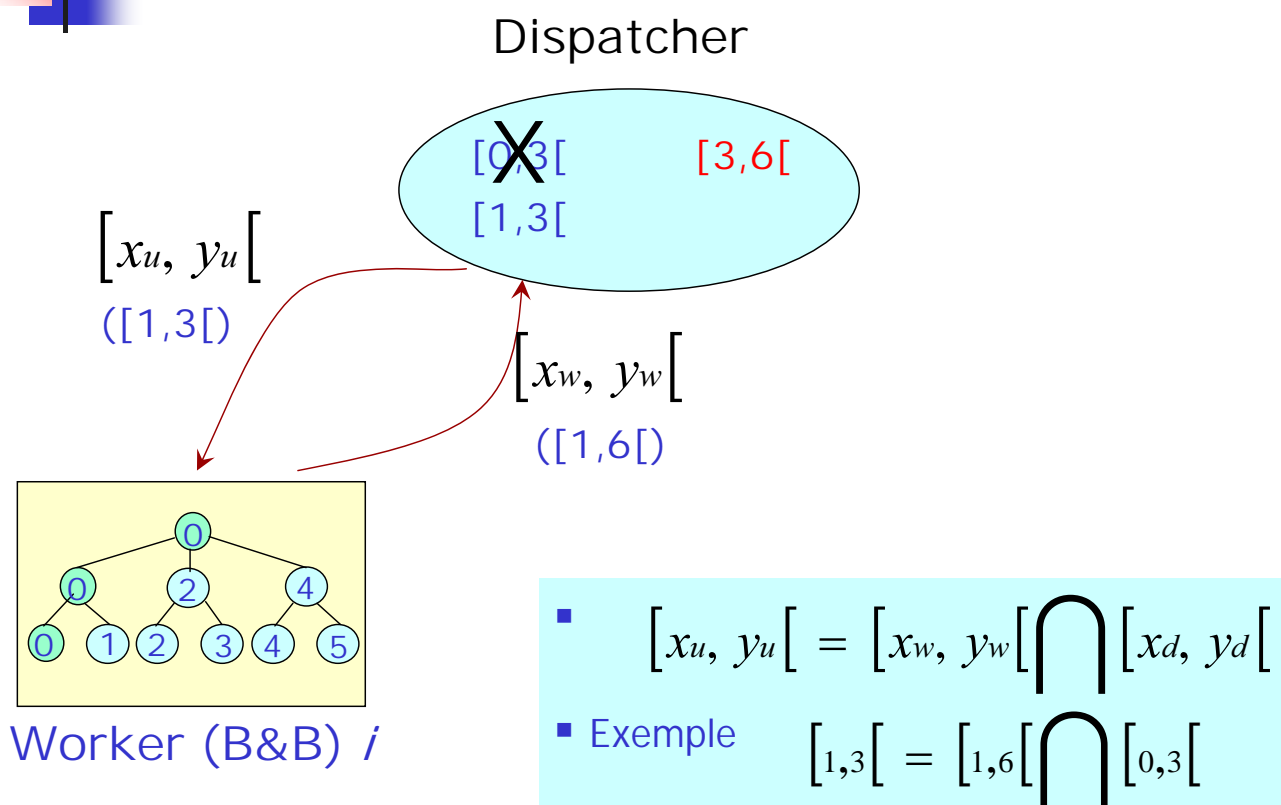
Dispatcher



Répartition de charge (2)

- Unité de répartition = intervalle
- Réception d'une demande de travail
 - Choisir le plus grand intervalle I
 - Si la taille de I est supérieure à un seuil S ,
 - donner la 2^{ème} moitié de I au processus demandeur,
 - et laisser la 1^{ère} moitié au processus détenteur de I .
 - Sinon, donner tout I au processus demandeur
 - Le seuil S : granularité du parallélisme
- Réception d'un travail
 - Transformer l'intervalle reçu en ensemble de nœuds à explorer
 - Besoin d'un opérateur **UNFOLD** pour le *dépliage* des intervalles

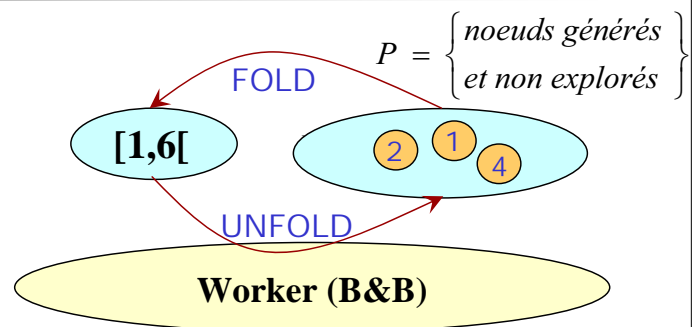
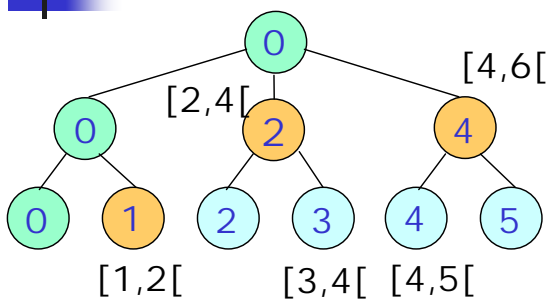
Checkpointing (1)



Checkpointing (2)

- Unité de *checkpoint* = intervalle
- Période de checkpoint = temps fixé
- Mise à jour régulière de chaque intervalle
 - La version du worker évolue durant l'exploration.
 - La version du dispatcher évolue après les opérations de répartition de charge.
- Transformation de l'ensemble de nœuds, restant à explorer, en intervalle
 - Besoin d'un opérateur FOLD pour *pliage* de nœuds

Opérateurs *FOLD* et *UNFOLD*



$$FOLD(\{1,2,4\}) = [1,2[\cup [2,4[\cup [4,6[= [1,6[$$

$$UNFOLD([1,6[) = \{1,2,4\}$$

- Opérateur *FOLD*

$$FOLD(P) = \bigcup_{n \in P} Int(n)$$

- Opérateur *UNFOLD*

$$UNFOLD([x, y[) = \{n, Int(n) \subseteq [x, y[\ \& \ Int(Père(n)) \not\subseteq [x, y[\}$$



Stratégie de partage de la *meilleure solution*

- Chaque worker **envoie au dispatcher** la meilleure solution (locale) trouvée ...
 - ... à la prochaine opération de checkpointing (sauvegarde)
 - ... si elle est meilleure que la meilleure solution (globale) connue
 - Chaque worker **récupère du dispatcher** la meilleure solution (globale) trouvée ...
 - ... à la prochaine opération de checkpointing (sauvegarde)
 - ... et remplace la meilleure solution (locale) si elle lui est supérieure
- + Limitation du nombre de messages échangés (pour le partage de la meilleure solution)



Détection de la terminaison

- Détection implicite et naturelle
- Chaque worker détecte la fin du calcul
 - ... en recevant un **intervalle nul** à sa demande de travail
 - ... et se retire de la grille
- Le dispatcher détecte la fin du calcul
 - ... lorsqu'il **n'a plus d'intervalle**



Implémentation de l'approche

- Langage de programmation C++
 - g++ : compilateur utilisé
 - STL : librairie utilisée
 - Appels RPC : communication worker/dispatcher
- Déploiement de l'application
 - Scripts shell
 - SSH : lancement et l'arrêt d'un calcul
- Aucune installation à faire sur les workers et le dispatcher
 - RPC, SSH installés par défaut.

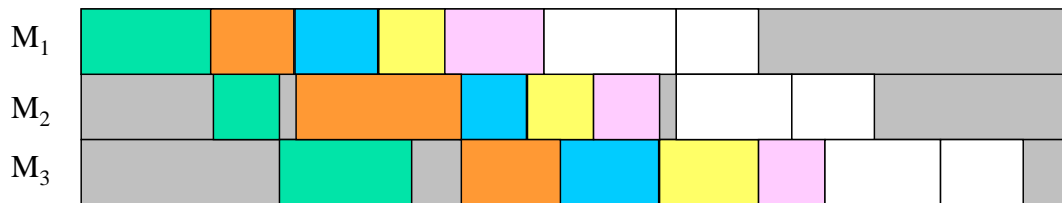


Plan

- Modèles parallèles pour les méthodes exactes
- Gridification de l'algorithme B&B
- Application au problème du Flow-Shop
- Conclusions et perspectives

Problème du Flow Shop

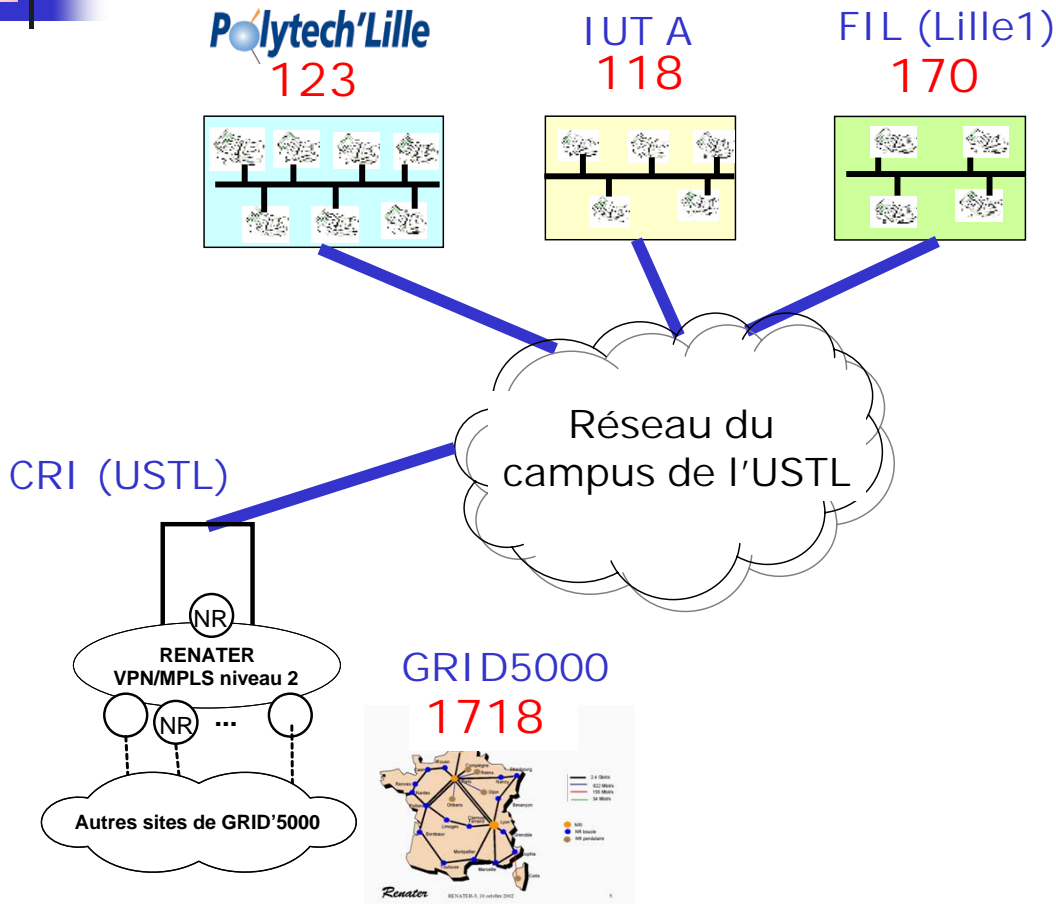
- Ordonnancement de N travaux sur M machines
 - Deux travaux ne peuvent pas être traités simultanément sur la même machine
 - L'ordre de passage des travaux est le même sur toutes les machines
- Objectif à minimiser
 - *Makespan* (C_{max}) : temps total de traitement des travaux



Résolution sur grille de Ta056

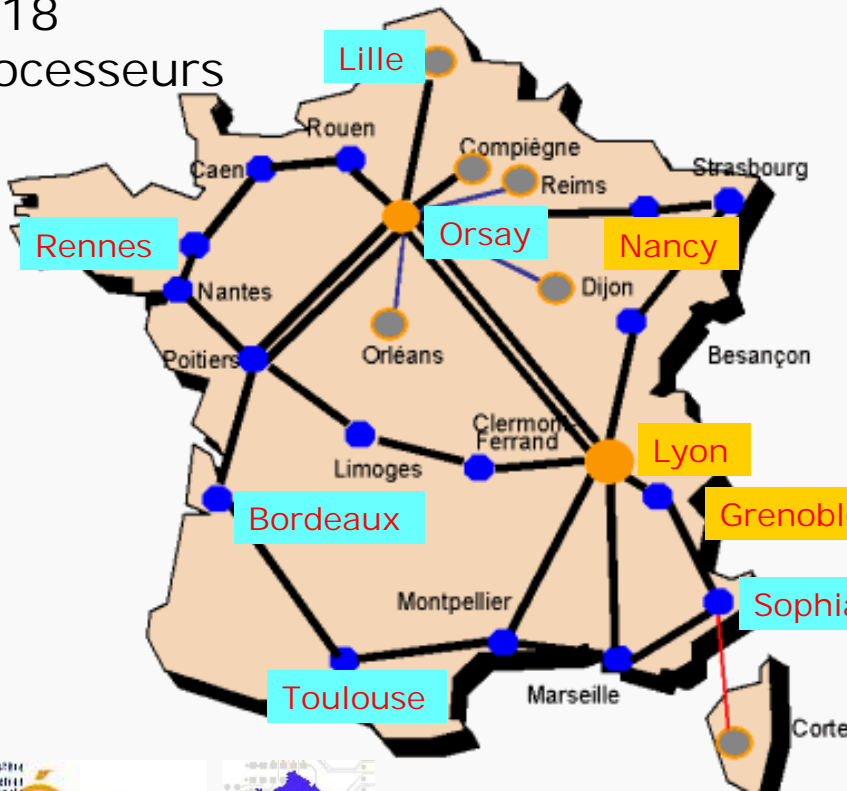
- Instances de Taillard
 - Instances de référence pour le Flow-Shop
 - EJOR 64, 1993
 - <http://ina2.eivd.ch/Collaborateurs/etd/default.htm>
- Instance résolue
 - Ta056 – 50 travaux sur 20 machines

Une grille de 2235 processeurs



Grid5000 (<http://www.grid5000.fr/>)

1718 processeurs



Universités



Conseils régionaux & généraux



Action Concertée Incitative [ACI]
Globalisation des Ressources Informatiques
et des Données [GRID]



Instance Ta056 du Flow-Shop 50-20 - Solution

- Meilleure borne inférieure
 - $C_{max} = 3367$
 - Vaessens, 1995
- Meilleure solution
 - $C_{max} = 3681$
 - Ruiz & Stutzle, 2004
- Solution exacte
 - $C_{max} = 3679$
 - Mezmaz, Melab & Talbi, 2006



Instance Ta056 du Flow-Shop 50-20 - Statistiques

Durée totale d'exécution	25 jours 46 minutes
Durée totale d'exécution agrégée	22 ans 185 jours 16h
Efficacité parallèle	97 %
Nombre de nœuds explorés	6,50874 e+12
Nombre de nœuds redondants	0,39 %
Taux d'utilisation CPU du dispatcher	1,7 %
Nombre moyen de processeurs utilisés	328
Pic de processeurs atteint sur toute la grille	1 195
Nombre de fois qu'un proc. a rejoint la grille	11 802
Pic de processeurs atteint sur chaque site	Bdx(88), Orsay(360), Sophia(190), Lille(98), Toulouse(112), Rennes(456), Univ.(304)
Nombre d'unités de travail (intervalles) alloué(e)s	129 958
Nombre d'opérations de sauvegarde	4 094 176



Autres défis réalisés

- Problème du Voyageur de Commerce
 - TSP/Usa13509 (8 Mai 1998)
 - <http://www.keck.caam.rice.edu/tsp/>
 - ~ 4 ans / ???
 - TSP/D15112 (20 Avril 2001)
 - <http://www.tsp.gatech.edu/d15sol/>
 - ~ 22 ans / Compaq EV6 Alpha 500 MHz
 - TSP/Sw24978 (mars 2003/mai 2004)
 - <http://www.tsp.gatech.edu/sweden/index.html>
 - ~ 84 ans / Intel Xeon 2.8 GHz
- Problème d'affectation quadratique
 - QAP/Nug30 (8 juin 2000 / 15 juin 2000)
 - <http://www-unix.mcs.anl.gov/metaneos/nug30/>
[BBC News, Le Monde, Chicago Sun Times, etc.](#)
 - ~ 7 ans / HP-C3000 400MHz
- Problème d'ordonnancement Flow-Shop
 - FS/Ta056 (9 octobre 2005 / 4 décembre 2005)
 - [Site Internet INRIA, et Grid5000](#)
 - ~ 22 ans / Temps cumulé



Plan

- Modèles parallèles pour les méthodes exactes
- Gridification de l'algorithme B&B
- Application au problème du Flow-Shop
- Conclusions et perspectives



Conclusion (1/3)

- Caractéristiques de la grille ...
 - ... volatilité, hétérogénéité, multi-domaine d'administration, et leur dimension (grande échelle)
- Caractéristiques du B&B
 - ... recherche d'une solution exacte
 - ... exploration d'un arbre irrégulier
 - ... partage de la meilleure solution trouvée
 - ... détection de la terminaison
- Re-penser l'algorithme pour prendre en compte
 - ... ses caractéristiques et celles des grilles de calcul



Conclusion (2/3)

- Approche basée sur un codage particulier de l'arbre de base et des unités de travail
 - Méthode de numérotation des nœuds de l'arbre de base
 - Codage d'un pool de nœuds par un simple intervalle
 - Extension du B&B par les opérateurs de *FOLD* et *UNFOLD*
- Répartition dynamique de la charge
 - Arbre irrégulier
 - dans un contexte volatile, hétérogène et large échelle
 - Unité de répartition = intervalle
 - Réduction du coût des communications
 - Résultats sur Ta056 : efficacité de 97%



Conclusion (3/3)

- Checkpointing
 - Recherche d'une solution exacte dans un contexte volatile et hétérogène
 - Unité de checkpoint = intervalle (données d'un B&B)
 - Réduction du coût de sauvegarde (communication réseau et stockage)
 - Résultat sur T056 : plus de 4 millions de sauvegarde
- Partage de la meilleure solution connue
 - Partage non explicite à coût négligeable
 - Solution communiquée dans les messages de sauvegarde
- Détection de la terminaison
 - Implicite, asynchrone et naturelle
 - Réception d'un intervalle vide par les workers, plus d'intervalle dans le dispatcher



Perspectives (1/2)

1. Généralisation de l'approche aux autres problèmes et stratégies de parcours
 - ... applicable aux problèmes de permutation
 - ... avec parcours de l'arbre en profondeur d'abord
2. Prise en compte de l'hétérogénéité
 - La taille de l'intervalle alloué doit être proportionnelle à la puissance des machines
 - ... Vol de cycles, plus de demandes de travail des machines les plus puissantes
3. Résolution d'autres problèmes
 - ... Q3AP – Projet CHOC de l'ANR « Calcul Intensif et Grilles de Calcul »



Perspectives (2/2)

4. Test de la limite de l'approche (passage à l'échelle)

- Avec pic de ~1200 processeurs,
- le taux d'utilisation CPU du dispatcher = 1,7%
- Goulet d'étranglement à quelle échelle ?

5. Proposition une approche pair-à-pair ...

- ... inspirée de l'architecture de Gnutella
- ... un index (associations *Intervalle – Adresse IP*)
- ... des relais (super peers) pour le passage de pare-feux
- ... des pairs (volontaires) pour l'exploration de l'arbre



Réserve



Définitions

- $\alpha (i)$ – Performance normalisée du worker i ($i \in I$),
- $U (i)$ – Durée cumulée de « disponibilité » du worker i ,
- $w (j)$ – Index du worker qui a accompli la tâche j ($j \in J$),
- $t (j)$ – Temps CPU consacré à accomplir la tâche j ,
- W – Temps total de disponibilité des workers $\sum_{i \in I} U (i)$
- T – Durée cumulée totale à l'exécution des tâches $\sum_{j \in J} t (j)$

Statistiques

- \mathcal{T} - Temps cumulé normalisé

$$\mathcal{T} = \sum_{j \in J} \alpha(w(j)) \times t(j)$$

- \mathcal{P} - Performance équivalente du pool

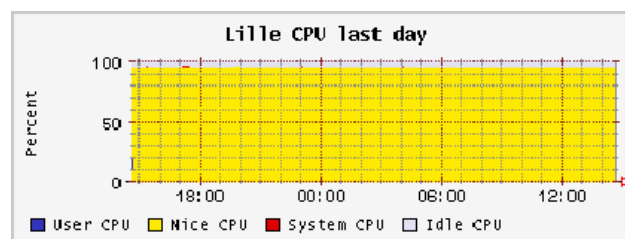
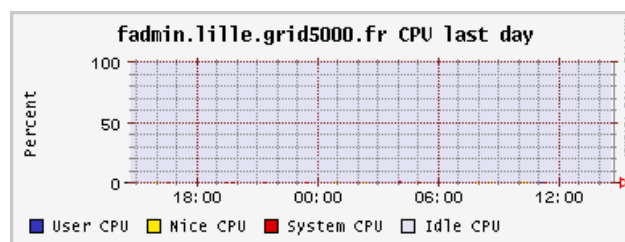
$$\mathcal{P} = \frac{\sum_{i \in I} \alpha(i) \times U(i)}{\sum_{i \in I} U(i)}$$

- \mathcal{N} - Nombre moyen de workers

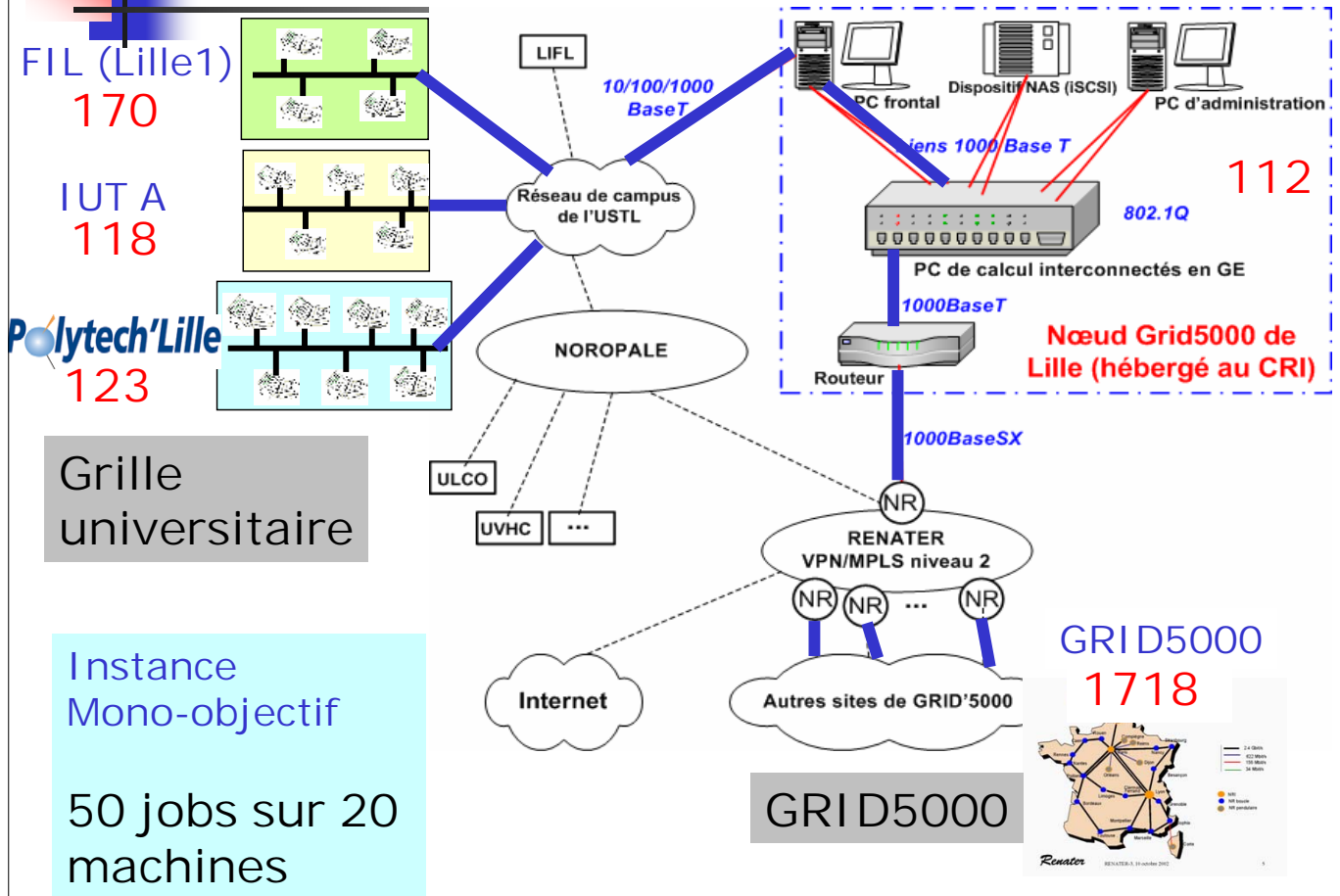
$$\mathcal{N} = \frac{\sum_{i \in I} U(i)}{W}$$

- n - Efficacité parallèle

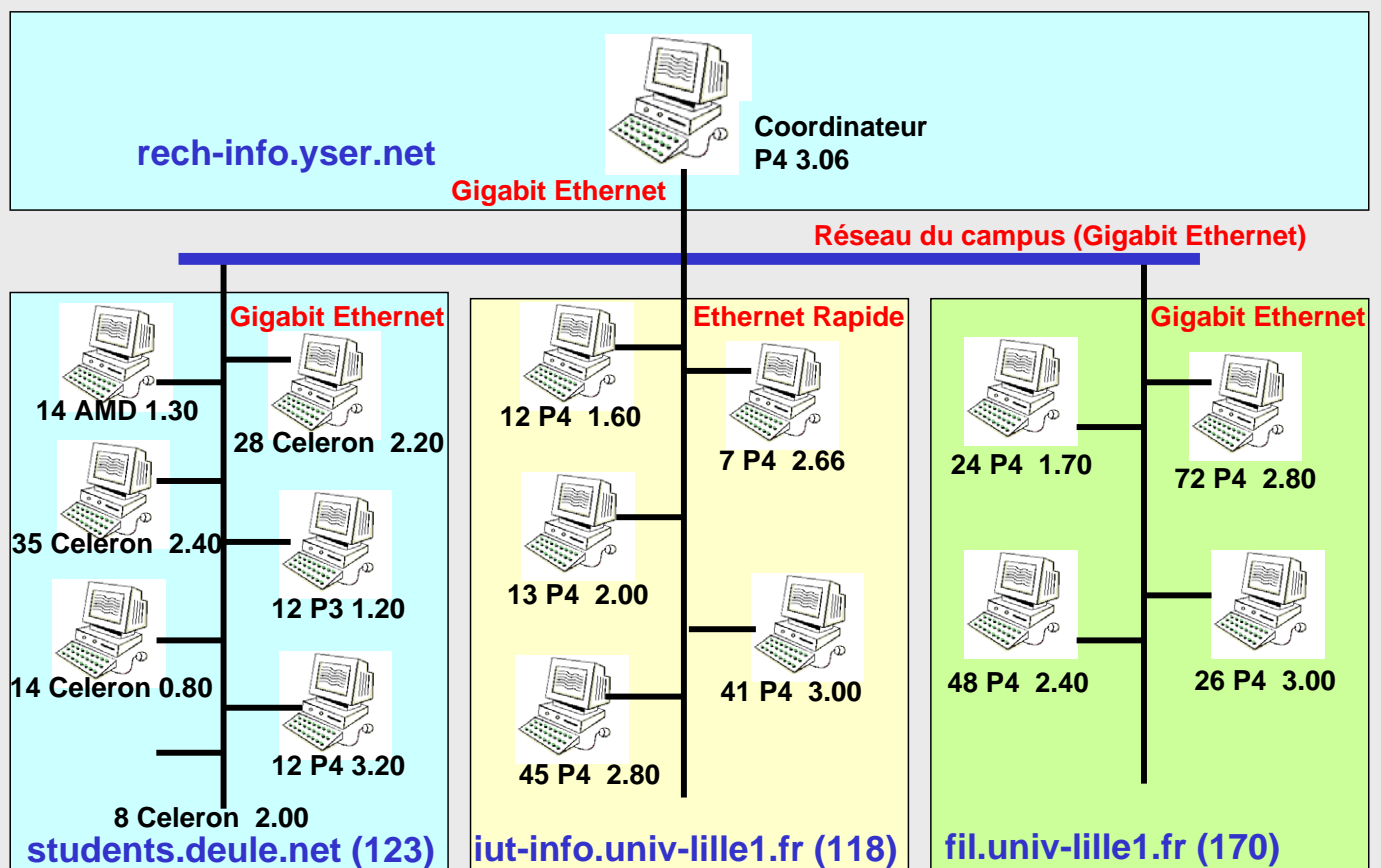
$$n = \frac{\sum_{j \in J} t(j)}{\sum_{i \in I} U(i)}$$



Une grille de 2235 processeurs

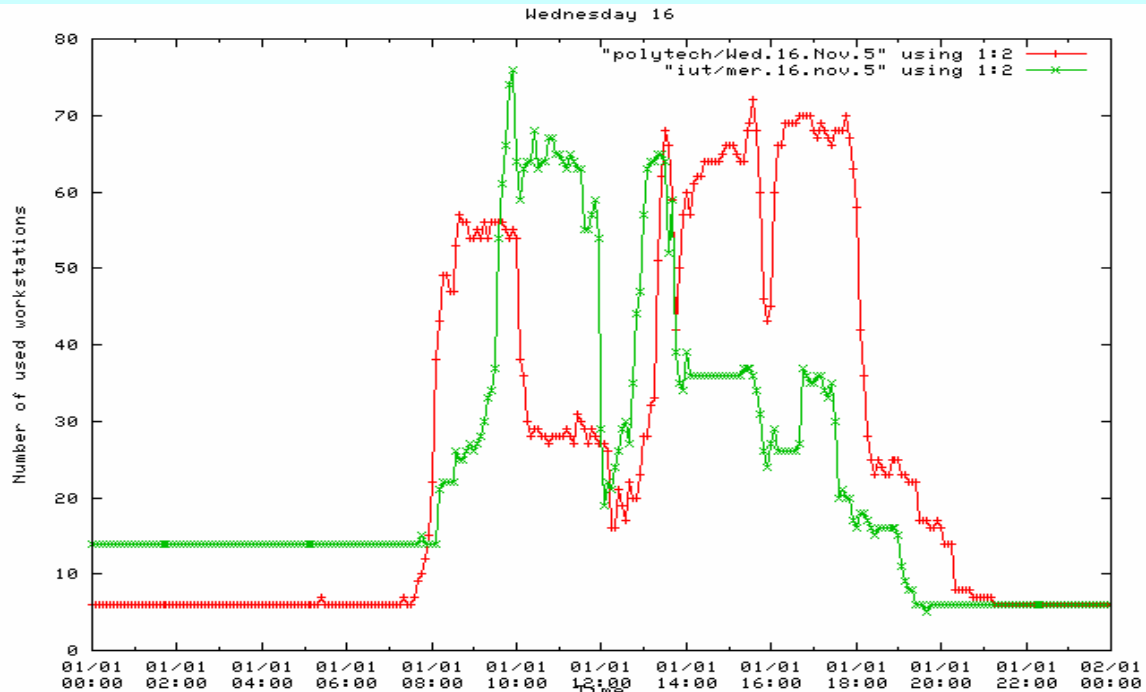


Grille universitaire (412 PC)



Temps de disponibilité des machines (1 semaine) 89% à Polytech-Lille et 88% à l'IUT-A

- Grille dédiée à l'enseignement informatique sous-exploitée (temps de disponibilité des machines : ~90%)
- Plate-forme potentiellement puissante à faible coût



Instance Ta056 du Flow-Shop 50-20 - Statistiques

Durée totale d'exécution	25 jours 46 minutes
Date de début	11 déc. 2005, 18h03
Date de fin	05 jan. 2006, 18h50
Durée totale d'exécution agrégée	22 ans 185 jours 16h
Nombre moyen de processeurs utilisés	328
Pic de processeurs atteint sur toute la grille	1 195
Nombre de fois qu'un proc. a rejoint la grille	11 802
Pic de processeurs atteint sur chaque site	Bdx(88), Orsay(360), Sophia(190), Lille(98), Toulouse(112), Rennes(456), Univ.(304)
Nombre de nœuds explorés	6,50874 e+12
Nombre de nœuds redondants	0,39 %
Unités de travail (intervalles) alloué(e)s	129 958
Opérations de sauvegarde	4 094 176
Efficacité parallèle	97 %
Taux d'utilisation CPU du dispatcher	1,7 %