Ordonnancement et réplication de données bioinformatiques dans un contexte de grille de calcul

Antoine Vernois

11 octobre 2006

École normale supérieure de Lyon – LIP

Institut de Biologie des Protéines

Plan de l'expos

Grands projets de séquencage de génomes complets

homme, souris, animaux, végétaux, unicellulaires

Croissance exponentielle de l'information biologique

- doublement de la taille des banques tous les 12 mois
- 1 ou 2 nouveaux génomes complets par mois
- plusieurs dizaines de Gos par mois

Traitements informatiques

- de courtes durées
- en très grand nombre

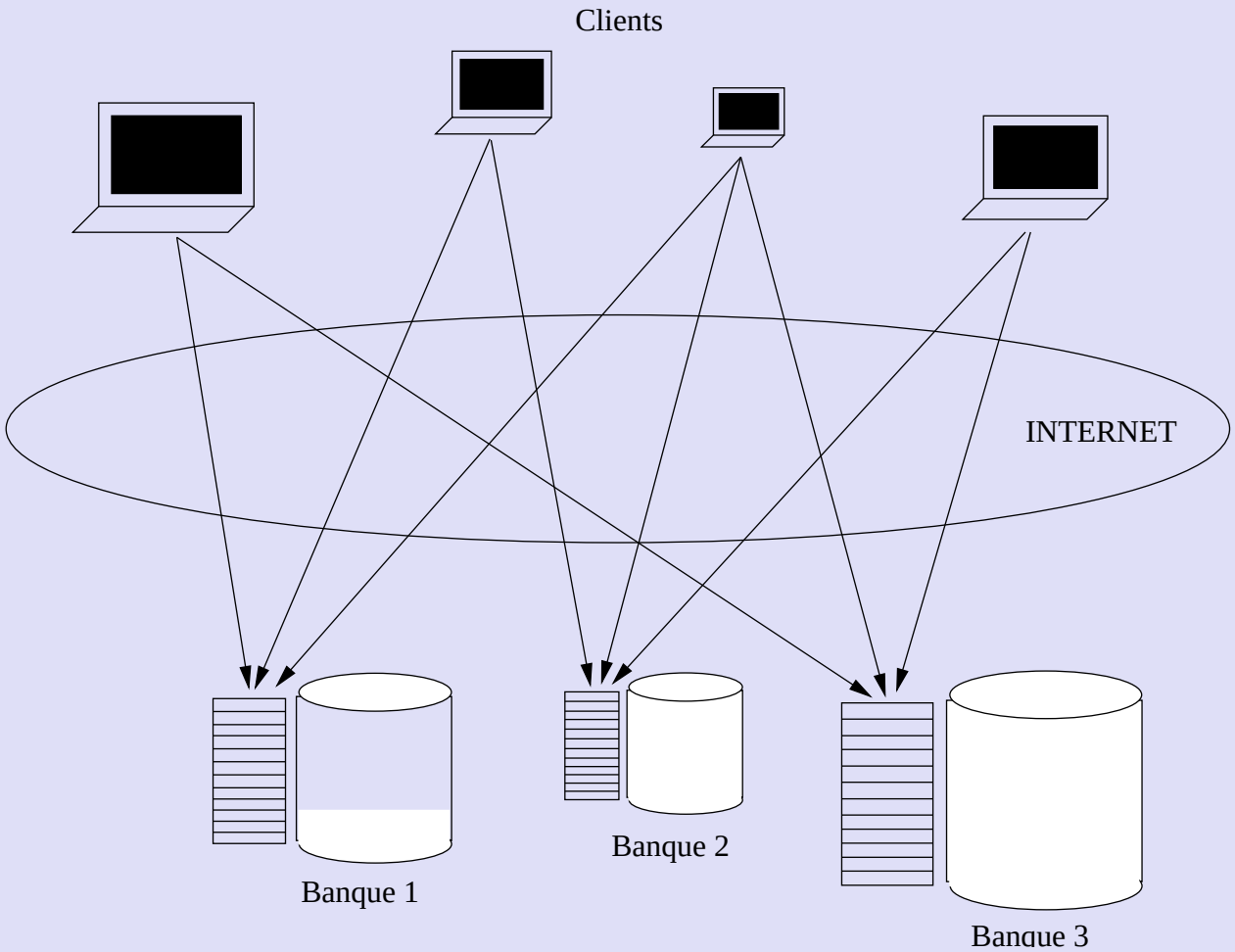
Des chiffres

Provenant de traces du serveur NPS@ de l'IBCP

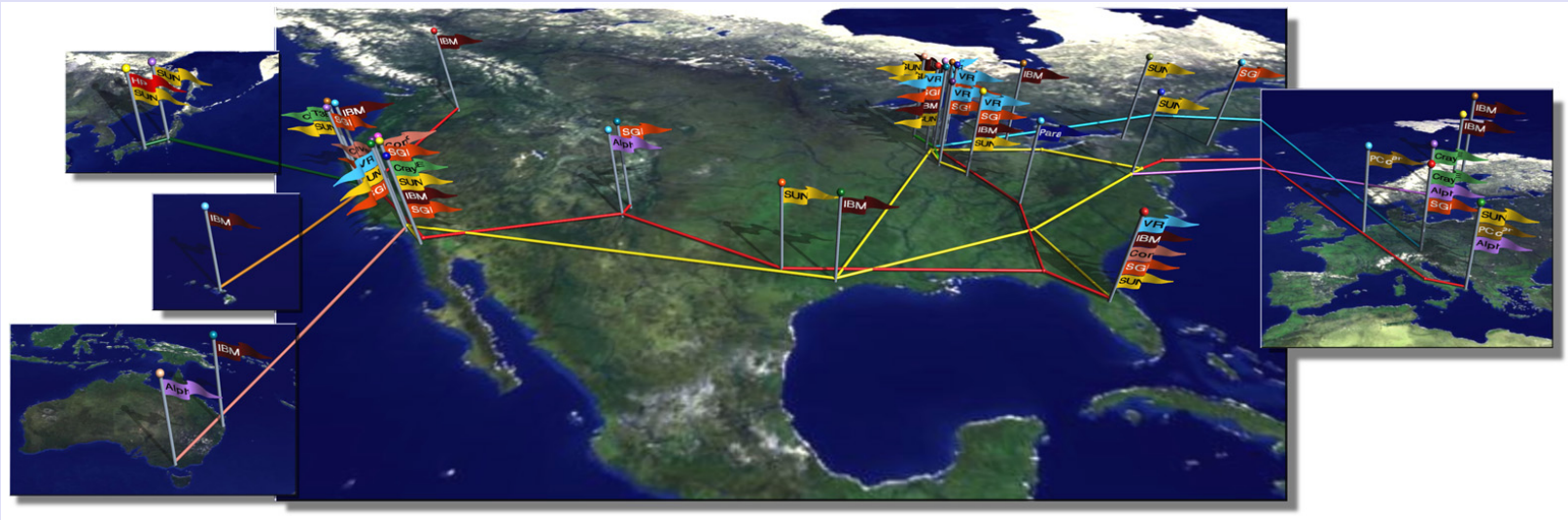
http://npsa-pbil.ibcp.fr/

nombre de banques de données	23
nombre d'algorithmes	8
nombre de couples algorithme-banque de données	80
nombre de requêtes	88730
taille de la plus petite banque	1 MB
taille de la plus grande banque	12 GB

La bioinformatique actuellement



Les grilles de calculs



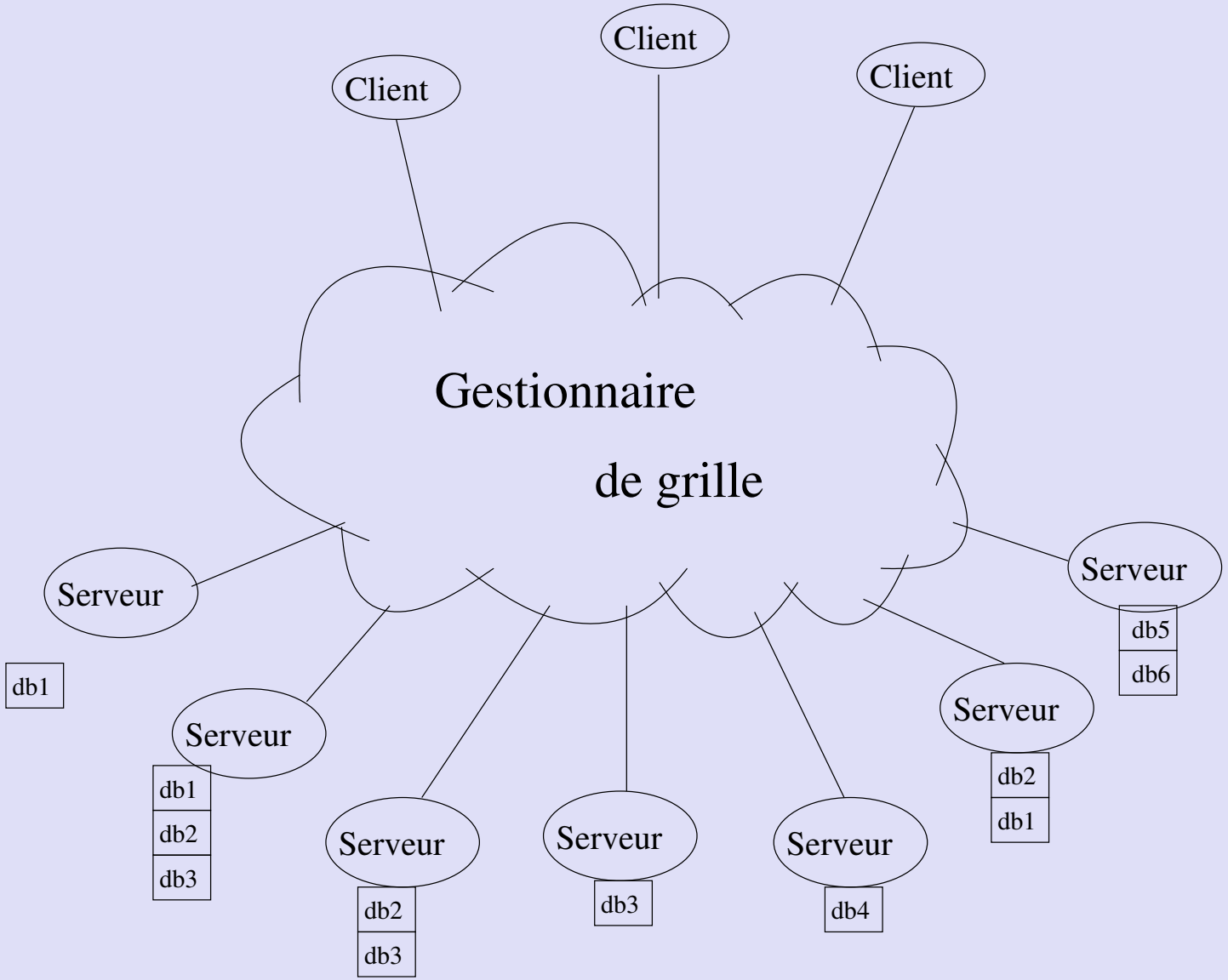
Une grille

- mutualisation de ressources
  - calcul
  - stockage
- accès unifié et transparent

Une solution possible

- pour le stockage
- pour les ressources de calcul
- pour l'accès à ces ressources

La bioinformatique et les grilles



L'idée simple

On place toutes les banques sur tous les serveurs

- pas possible : trop de données
- beaucoup de gâchis d'espace
  - toutes les banques ne sont pas utilisées tout le temps
  - (embl.fas = 12GO, $< 1\%$ des requêtes)
- les mises à jour deviennent coûteuses

Cache-Web ?

Similitude avec les pages web. Mais

- échelles différentes

L'idée simple

On place toutes les banques sur tous les serveurs

- pas possible : trop de données
- beaucoup de gâchis d'espace
  - toutes les banques ne sont pas utilisées tout le temps
  - (embl.fas = 12GO, $< 1\%$ des requêtes)
- les mises à jour deviennent coûteuses

Cache-Web ?

Similitude avec les pages web. Mais

- échelles différentes
- coûts de calcul

L'idée simple

On place toutes les banques sur tous les serveurs

- pas possible : trop de données
- beaucoup de gâchis d'espace
  - toutes les banques ne sont pas utilisées tout le temps
  - (embl.fas = 12GO, $< 1\%$ des requêtes)
- les mises à jour deviennent coûteuses

Cache-Web ?

Similitude avec les pages web. Mais

- échelles différentes
- coûts de calcul

L'idée simple

On place toutes les banques sur tous les serveurs

- pas possible : trop de données
- beaucoup de gâchis d'espace
  - toutes les banques ne sont pas utilisées tout le temps
  - (embl.fas = 12GO, $< 1\%$ des requêtes)
- les mises à jour deviennent coûteuses

Cache-Web ?

Similitude avec les pages web. Mais

- échelles différentes
- coûts de calcul

L'idée simple

On place toutes les banques sur tous les serveurs

- pas possible : trop de données
- beaucoup de gâchis d'espace
  - toutes les banques ne sont pas utilisées tout le temps
  - (embl.fas = 12GO, $< 1\%$ des requêtes)
- les mises à jour deviennent coûteuses

Cache-Web ?

Similitude avec les pages web. Mais

- échelles différentes
- coûts de calcul

Travaux précédents

Service de réplication

- mécanismes pour localisation, stockage, réplication
  - souvent à la charge de l'utilisateur
- ne tient pas compte de tous les paramètres
  - capacité de stockage et calcul
  - utilisation des données
  - type d'opérations

Réplication et ordonnancement

- Foster et al.
  - décision locale de réplication
  - ordonnancement externe sur les sites ayants les données
- Chakrabarti et al. : IRS
  - ordonnancement « on-line »
  - placement amélioré itérativement en fonction de l'utilisation

Travaux précédents

Service de réplication

- mécanismes pour localisation, stockage, réplication
  - souvent à la charge de l'utilisateur
- ne tient pas compte de tous les paramètres
  - capacité de stockage et calcul
  - utilisation des données
  - type d'opérations

Réplication et ordonnancement

- Foster et al.
  - décision locale de réplication
  - ordonnancement externe sur les sites ayants les données
- Chakrabarti et al. : IRS
  - ordonnancement « on-line »
  - placement amélioré itérativement en fonction de l'utilisation

Travaux précédents

Service de réplication

- mécanismes pour localisation, stockage, réplication
  - souvent à la charge de l'utilisateur
- ne tient pas compte de tous les paramètres
  - capacité de stockage et calcul
  - utilisation des données
  - type d'opérations

Réplication et ordonnancement

- Foster et al.
  - décision locale de réplication
  - ordonnancement externe sur les sites ayants les données
- Chakrabarti et al. : IRS
  - ordonnancement « on-line »
  - placement amélioré itérativement en fonction de l'utilisation

Analyse des traces

À partir des logs de NPS@ :

- des utilisations différentes entre les requêtes
  - BLAST sur Swiss-Prot : 77% des requêtes
- globalement, l'utilisation reste la même

On peut utiliser l'analyse des requêtes précédentes comme point de départ.

Analyse des traces

À partir des logs de NPS@ :

- des utilisations différentes entre les requêtes
  - BLAST sur Swiss-Prot : 77% des requêtes
- globalement, l'utilisation reste la même

On peut utiliser l'analyse des requêtes précédentes comme point de départ.

Analyse des traces

À partir des logs de NPS@ :

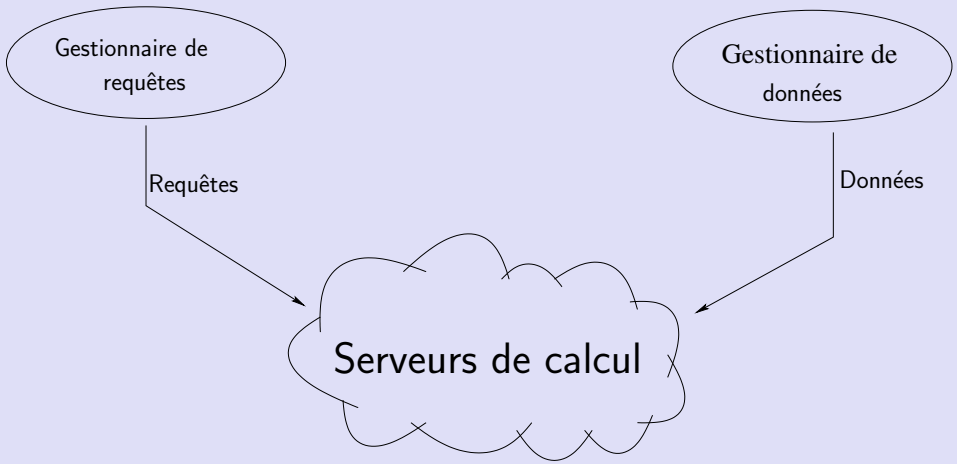
- des utilisations différentes entre les requêtes
  - BLAST sur Swiss-Prot : 77% des requêtes
- globalement, l'utilisation reste la même

On peut utiliser l'analyse des requêtes précédentes comme point de départ.

Sommaire

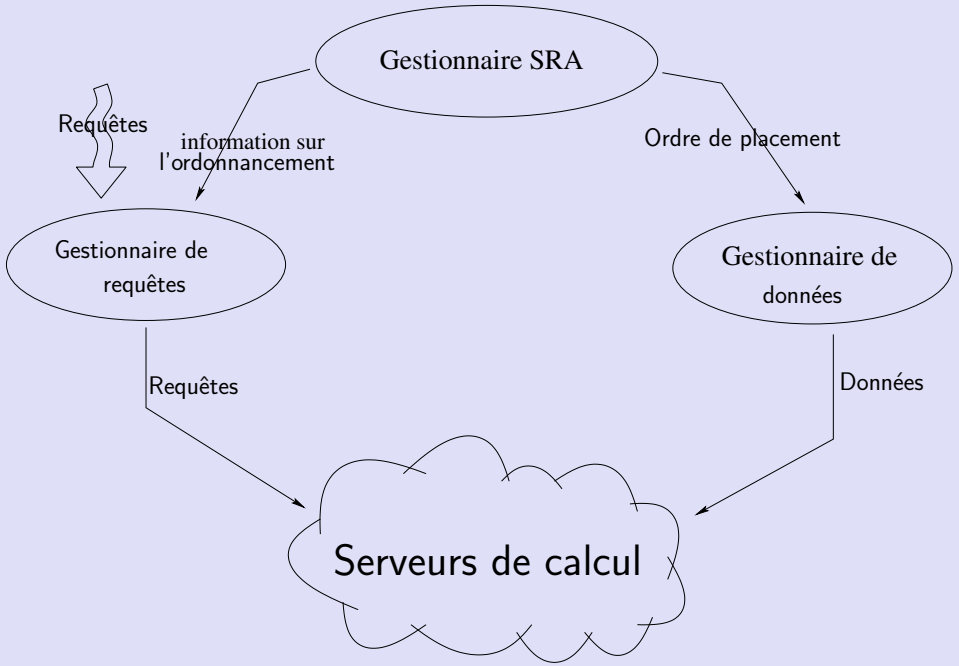
Plan de l'expos

Intégration dans une plate-forme



- Calcul du placement et de l'ordonnancement
- Placement des données sur les serveurs
- Ordonnancement des requêtes à leur arrivée

Intégration dans une plate-forme



- Calcul du placement et de l'ordonnancement
- Placement des données sur les serveurs
- Ordonnancement des requêtes à leur arrivée

Modélisation - 1/2

La plate-forme

- n serveurs de calcul P_i
  - espace de stockage : m_i
  - puissance de calcul : w_i

Les données bioinformatiques

- m banques de données d_j de taille : $size_j$
- p algorithmes a_k :
  - coût linéaire : $\alpha_k.size_j + c_k$
- des requêtes $R(k, j)$
  - fréquence d'utilisation du couple : $f(k, j)$

Modélisation - 1/2

La plate-forme

- n serveurs de calcul P_i
  - espace de stockage : m_i
  - puissance de calcul : w_i

Les données bioinformatiques

- m banques de données d_j de taille : $size_j$
- p algorithmes a_k :
  - coût linéaire : $\alpha_k.size_j + c_k$
- des requêtes $R(k, j)$
  - fréquence d'utilisation du couple : $f(k, j)$

Modélisation - 1/2

La plate-forme

- n serveurs de calcul P_i
  - espace de stockage : m_i
  - puissance de calcul : w_i

Les données bioinformatiques

- m banques de données d_j de taille : $size_j$
- p algorithmes a_k :
  - coût linéaire : $\alpha_k.size_j + c_k$
- des requêtes $R(k, j)$
  - fréquence d'utilisation du couple : $f(k, j)$

Modélisation - 2/2

Les inconnues

- δ_i^j : placement de la donnée d_j sur P_i
- $n_i(k, j)$: nombre de requêtes de type $R(k, j)$ effectuées sur P_i
- TP : débit de la plate-forme en terme de requêtes

Les contraintes 1/2

Les contraintes sur le placement

- toutes les données stockées au moins une fois

$$\forall i \in [1, m], \sum_{j=1}^{n} \delta_i^j \geqslant 1$$

- un serveur ne peut pas stocker plus que son espace

$$\forall i \in [1, m], \sum_{j=1}^{n} \delta_i^j .size_j \leqslant m_i$$

Les contraintes sur le calcul

Les contraintes 1/2

Les contraintes sur le placement

- toutes les données stockées au moins une fois

$$\forall i \in [1, m], \sum_{j=1}^{n} \delta_i^j \geqslant 1$$

- un serveur ne peut pas stocker plus que son espace

$$\forall i \in [1, m], \sum_{j=1}^{n} \delta_i^j . size_j \leqslant m_i$$

Les contraintes sur le calcul

- un serveur ne peut pas exécuter plus que sa puissance

$$\forall i \in [1, m], \sum_{j=1}^{n} \delta_i^j . load_j \leqslant p_i$$

Les contraintes 1/2

Les contraintes sur le placement

- toutes les données stockées au moins une fois

$$\forall i \in [1, m], \sum_{j=1}^{n} \delta_i^j \geqslant 1$$

- un serveur ne peut pas stocker plus que son espace

$$\forall i \in [1, m], \sum_{j=1}^{n} \delta_i^j . size_j \leqslant m_i$$

Les contraintes sur le calcul

- un serveur ne peut pas exécuter plus que sa puissance

$$\forall i \in [1, m], \sum_{k=1}^{p} \sum_{j=1}^{n} r_i(k, j)(\alpha_k * size_j + c_k) \leqslant w_i$$

Les contraintes 1/2

Les contraintes sur le placement

- toutes les données stockées au moins une fois

$$\forall i \in [1, m], \sum_{j=1}^{n} \delta_i^j \geqslant 1$$

- un serveur ne peut pas stocker plus que son espace

$$\forall i \in [1, m], \sum_{j=1}^{n} \delta_i^j . size_j \leqslant m_i$$

Les contraintes sur le calcul

- un serveur ne peut pas exécuter plus que sa puissance

$$\forall i \in [1, m], \sum_{k=1}^{p} \sum_{j=1}^{n} n_i(k, j)(\alpha_k * size_j + c_k) \leqslant w_i$$

Les contraintes 1/2

Les contraintes sur le placement

- toutes les données stockées au moins une fois

$$\forall i \in [1, m], \sum_{j=1}^{n} \delta_i^j \geqslant 1$$

- un serveur ne peut pas stocker plus que son espace

$$\forall i \in [1, m], \sum_{j=1}^{n} \delta_i^j . size_j \leqslant m_i$$

Les contraintes sur le calcul

- un serveur ne peut pas exécuter plus que sa puissance

$$\forall i \in [1, m], \sum_{k=1}^{p} \sum_{j=1}^{n} n_i(k, j)(\alpha_k * size_j + c_k) \leqslant w_i$$

Les contraintes 2/2

Les contraintes conjointes

- on exécute une requête que si la donnée est présente

$$\forall i \in [1,m], \forall j \in [1,n], \forall k \in [1,p], n_i(k,j) \leqslant v_{k,j}.\delta_i^j \cdot \frac{w_i}{\alpha_k.size_j + c_k}$$

- le nombre de jobs exécutés respecte les fréquences

$$\forall i \in [1,m], \forall j \in [1,n], \sum_{i=1}^{m} n_i(k,j) = f_{k,j}.TP$$

Objectif

Maximiser le débit de la plate-forme

Les contraintes 2/2

Les contraintes conjointes

- on exécute une requête que si la donnée est présente

$$\forall i \in [1,m], \forall j \in [1,n], \forall k \in [1,p], n_i(k,j) \leqslant v_{k,j}.\delta_i^j.\frac{w_i}{\alpha_k.size_j+c_k}$$

- le nombre de jobs exécutés respecte les fréquences

$$\forall i \in [1,m], \forall j \in [1,n], \sum_{i=1}^{m} n_i(k,j) = f_{k,j}.TP$$

Objectif

Maximiser le débit de la plate-forme

Les contraintes 2/2

Les contraintes conjointes

- on exécute une requête que si la donnée est présente

$$\forall i \in [1,m], \forall j \in [1,n], \forall k \in [1,p], n_i(k,j) \leqslant v_{k,j}.\delta_i^j.\frac{w_i}{\alpha_k.size_j+c_k}$$

- le nombre de jobs exécutés respecte les fréquences

$$\forall i \in [1,m], \forall j \in [1,n], \sum_{i=1}^{m} n_i(k,j) = f_{k,j}.TP$$

Objectif

Maximiser le débit de la plate-forme

Les contraintes 2/2

Les contraintes conjointes

- on exécute une requête que si la donnée est présente

$$\forall i \in [1,m], \forall j \in [1,n], \forall k \in [1,p], n_i(k,j) \leqslant v_{k,j}.\delta_i^j.\frac{w_i}{\alpha_k.size_j+c_k}$$

- le nombre de jobs exécutés respecte les fréquences

$$\forall i \in [1,m], \forall j \in [1,n], \sum_{i=1}^{m} n_i(k,j) = f_{k,j}.TP$$

Objectif

Maximiser le débit de la plate-forme

Le programme linéaire

Formulation du programme linéaire

MAXIMISER TP,
SOUS LES CONTRAINTES

$$
\begin{cases}
(1) \ \sum_{j=1}^{n} \delta_i^j \geqslant 1 & 1 \leqslant i \leqslant m \\
(2) \ \sum_{j=1}^{n} \delta_i^j . size_j \leqslant m_i & 1 \leqslant i \leqslant m \\
(3) \ \sum_{k=1}^{p} \sum_{j=1}^{n} n_i(k,j)(\alpha_k * size_j + c_k) \leqslant w_i & 1 \leqslant i \leqslant m \\
(4) \ n_i(k,j) \leqslant v_{k,j} . \delta_i^j . \dfrac{w_i}{\alpha_k . size_j + c_k} & 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n, 1 \leqslant k \leqslant p \\
(5) \ \sum_{i=1}^{m} n_i(k,j) = f_{k,j} . TP & 1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n
\end{cases}
$$

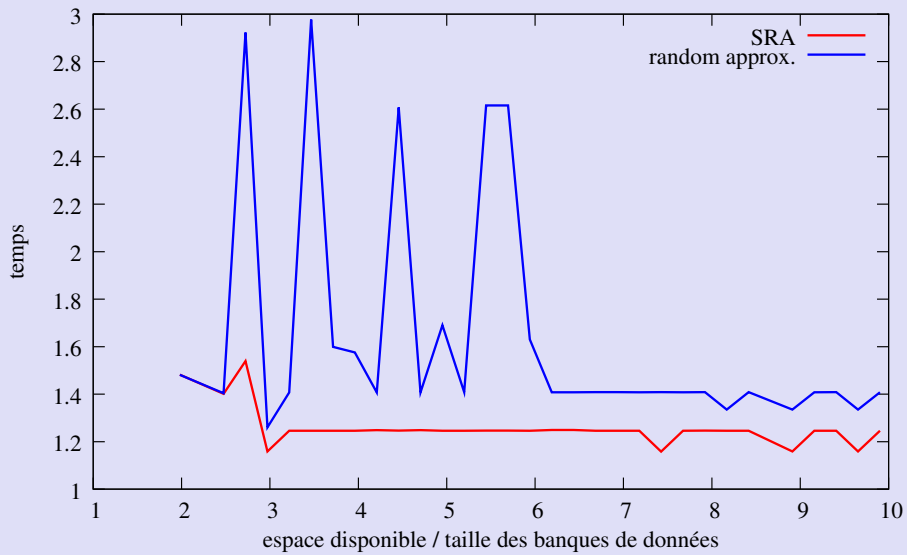
Approximer la solution

Complétude

Le problème d'ordonnancement et de placement conjoint est NP-complet.

Programme linéaire mixte

- approximation itérative entière pour δ_i^j



Simulation

Utilisation de OptorSim

- simulateur dédié à la gestion des données sur les grilles
- développé dans le cadre de EDG

Largement modifié pour répondre à nos besoins

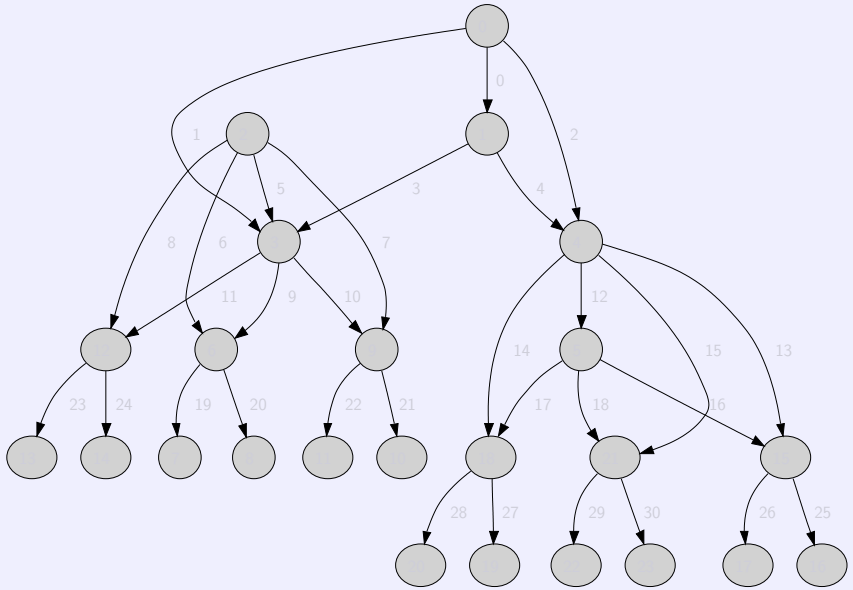
- hétérogénéité des serveurs de calculs
- CE type gestionnaire de batch
- intégration des temps de calculs
- mécanismes nécessaires à notre solution

Tests

Plate-forme simulée

- générée par Tiers
- 10 plate-formes

Exemple de plate-forme



Requêtes

- basées sur des requêtes réelles
- 40000 requêtes

Comparaison

- SRA : notre algorithme
- MCT : placement de SRA + ordonnancement MCT
- greedy : un placement glouton et ordonnancement MCT

Tests

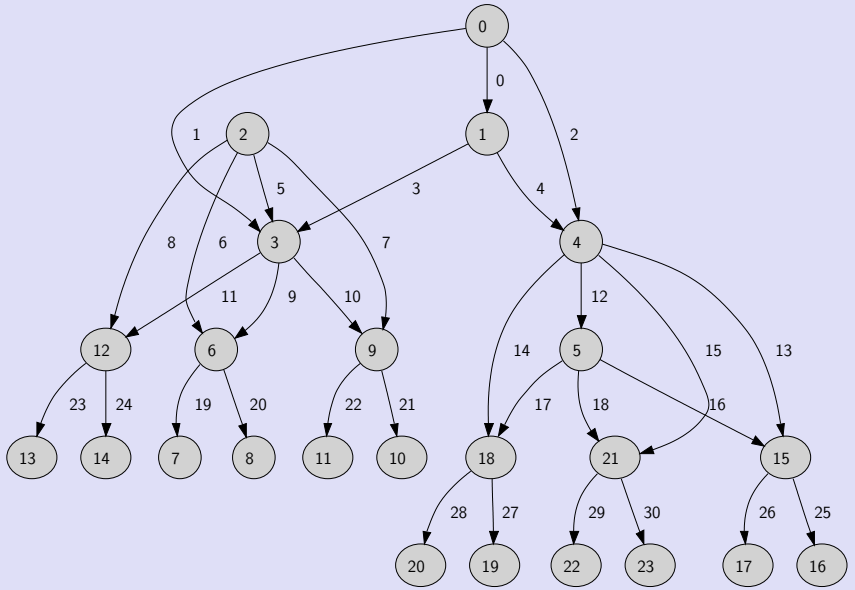
Plate-forme simulée

- générée par Tiers
- 10 plate-formes

Requêtes

- basées sur des requêtes réelles
- 40000 requêtes

Exemple de plate-forme



Comparaison

- SRA : notre algorithme
- MCT : placement de SRA + ordonnancement MCT
- greedy : un placement glouton et ordonnancement MCT

Tests

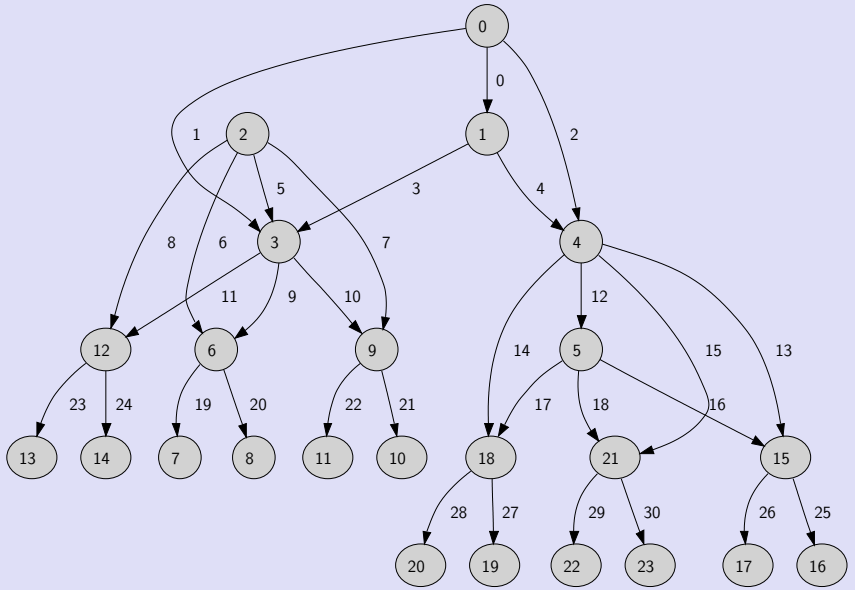
Plate-forme simulée

- générée par Tiers
- 10 plate-formes

Requêtes

- basées sur des requêtes réelles
- 40000 requêtes

Exemple de plate-forme



Comparaison

- SRA : notre algorithme
- MCT : placement de SRA + ordonnancement MCT
- greedy : un placement glouton et ordonnancement MCT

Tests

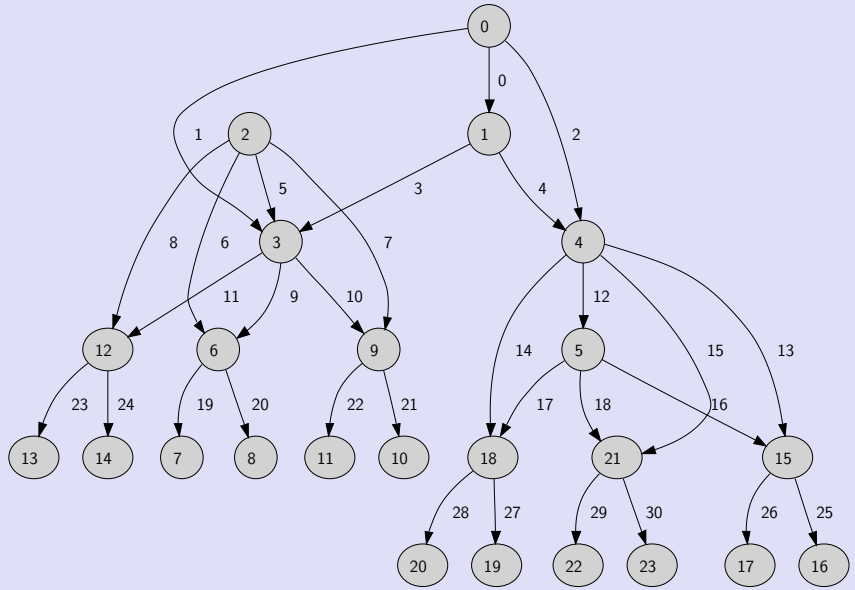
Plate-forme simulée

- générée par Tiers
- 10 plate-formes

Requêtes

- basées sur des requêtes réelles
- 40000 requêtes

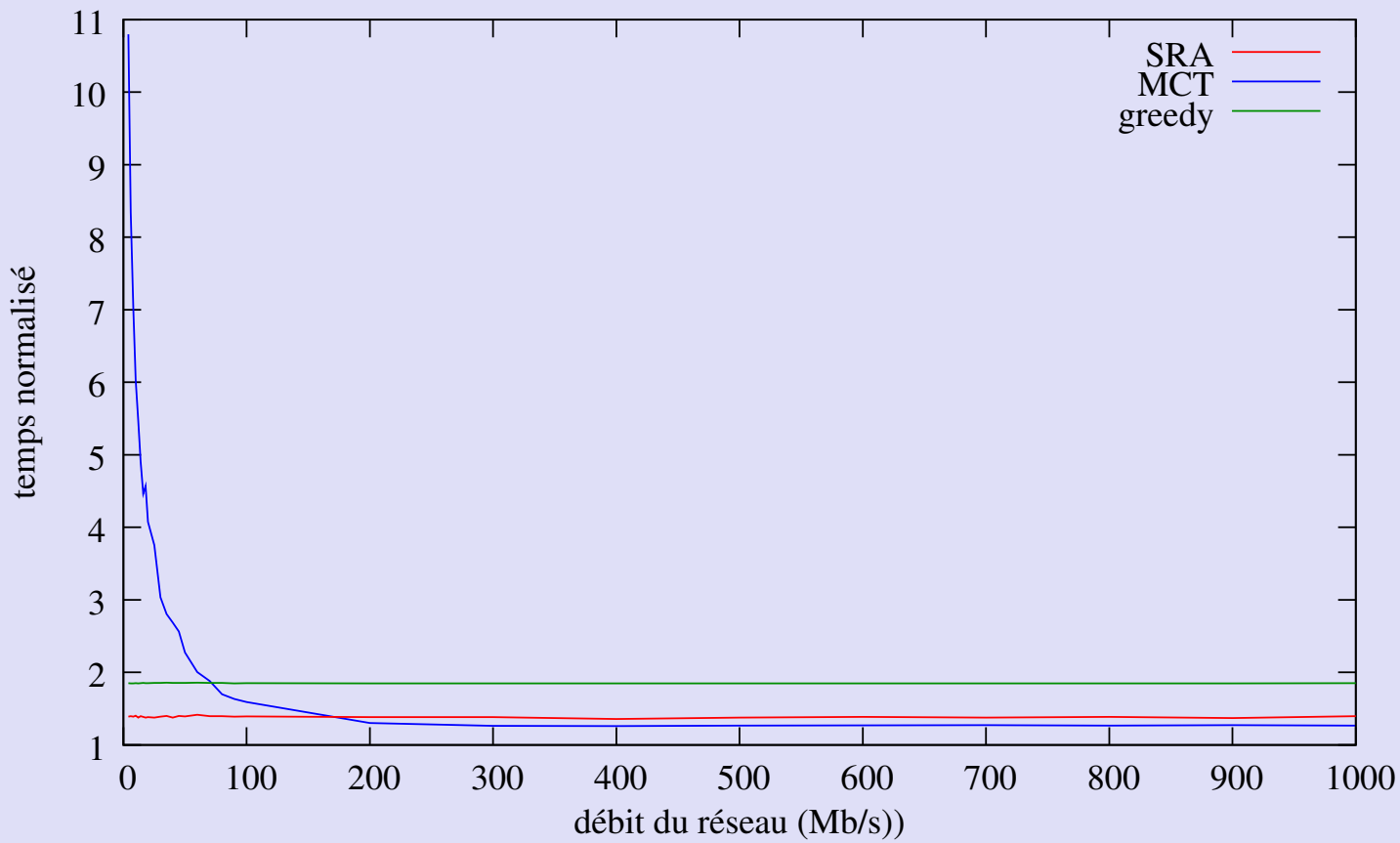
Exemple de plate-forme



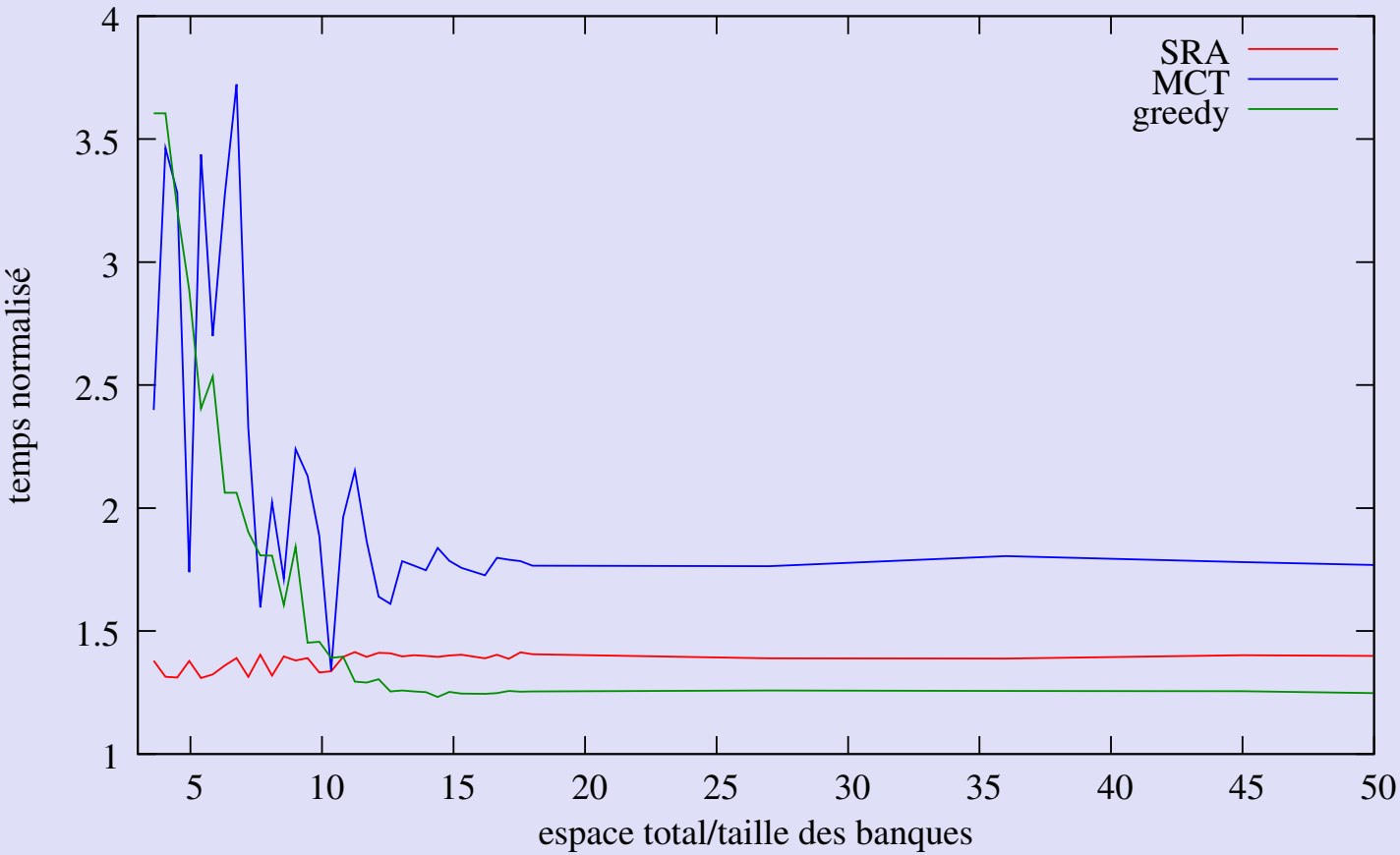
Comparaison

- SRA : notre algorithme
- MCT : placement de SRA + ordonnancement MCT
- greedy : un placement glouton et ordonnancement MCT

Temps d'exécution : fonction du débit réseau



Temps d'exécution : fonction de l'espace de stockage



Implémentation d'un prototype

Plate-forme logicielle

- DIET : Distributed Interactive Engineering Toolbox
  - gestionnaire de requête
  - LIP - ENS Lyon
- DTM : Data Tree Manager
  - gestionnaire de données pour la grille
  - Université de Besançon
- LogService + VizDIET
  - monitoring et visualisation

Plate-forme matérielle

- Grid'5000
  - 11 serveurs
  - 5 sites

Implémentation d'un prototype

Plate-forme logicielle

- DIET : Distributed Interactive Engineering Toolbox
  - gestionnaire de requête
  - LIP - ENS Lyon
- DTM : Data Tree Manager
  - gestionnaire de données pour la grille
  - Université de Besançon
- LogService + VizDIET
  - monitoring et visualisation

Plate-forme matérielle

- Grid'5000
  - 11 serveurs
  - 5 sites

Implémentation d'un prototype

Plate-forme logicielle

- DIET : Distributed Interactive Engineering Toolbox
  - gestionnaire de requête
  - LIP - ENS Lyon
- DTM : Data Tree Manager
  - gestionnaire de données pour la grille
  - Université de Besançon
- LogService + VizDIET
  - monitoring et visualisation

Plate-forme matérielle

- Grid'5000
  - 11 serveurs
  - 5 sites

Implémentation d'un prototype

Plate-forme logicielle

- DIET : Distributed Interactive Engineering Toolbox
  - gestionnaire de requête
  - LIP - ENS Lyon
- DTM : Data Tree Manager
  - gestionnaire de données pour la grille
  - Université de Besançon
- LogService + VizDIET
  - monitoring et visualisation

Plate-forme matérielle

- Grid'5000
  - 11 serveurs
  - 5 sites

Implémentation d'un prototype

Plate-forme logicielle

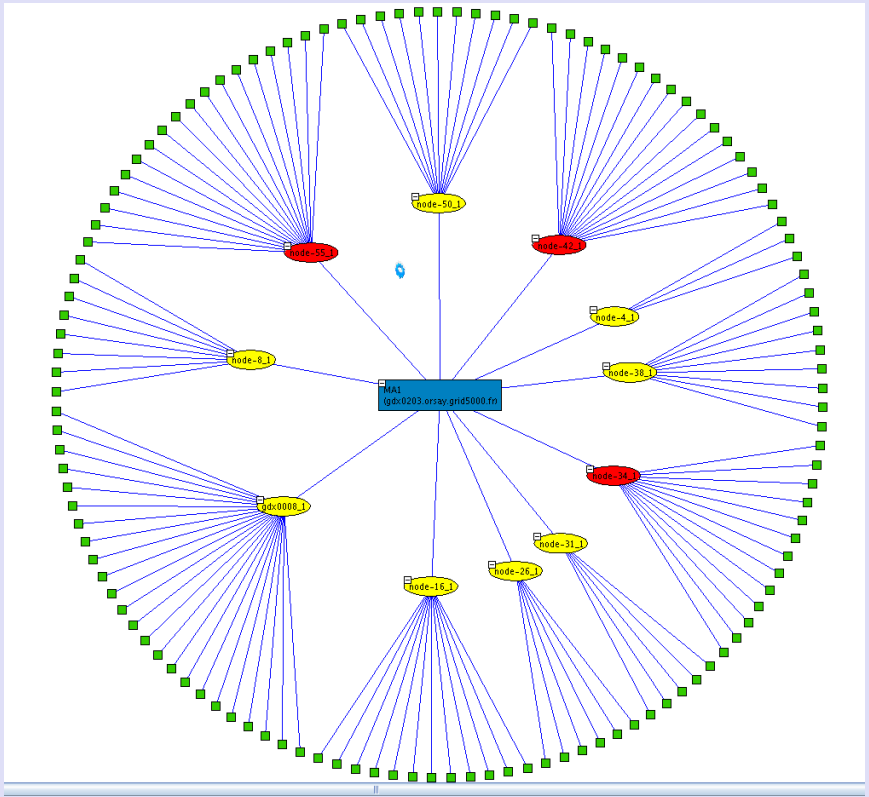
- DIET : Distributed Interactive Engineering Toolbox
  - gestionnaire de requête
  - LIP - ENS Lyon
- DTM : Data Tree Manager
  - gestionnaire de données pour la grille
  - Université de Besançon
- LogService + VizDIET
  - monitoring et visualisation

Plate-forme matérielle

- Grid'5000
  - 11 serveurs
  - 5 sites

Répartitions des données

Greedy



SRA

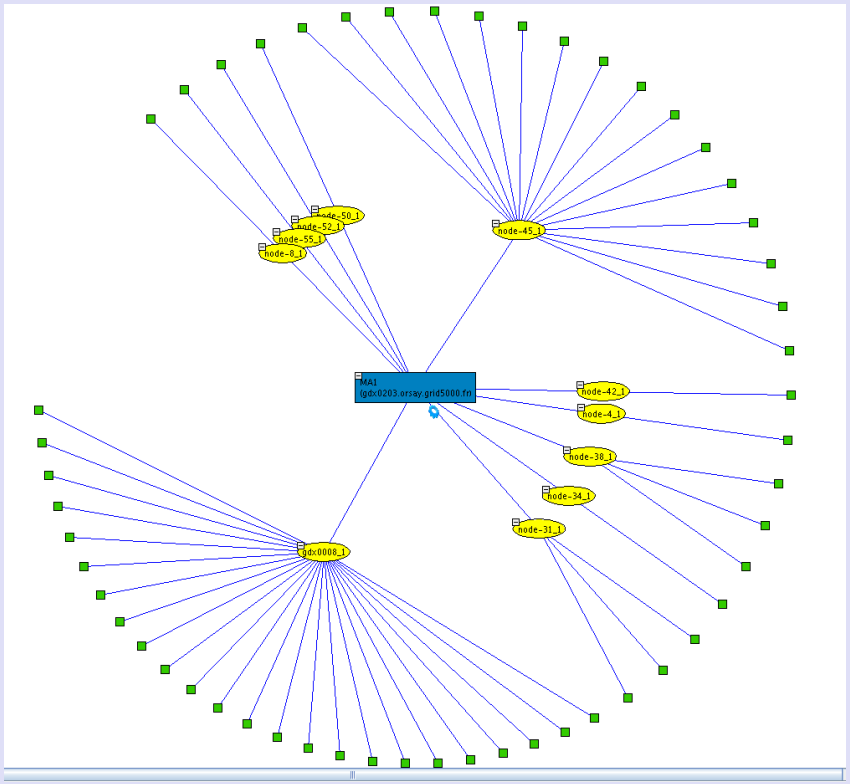
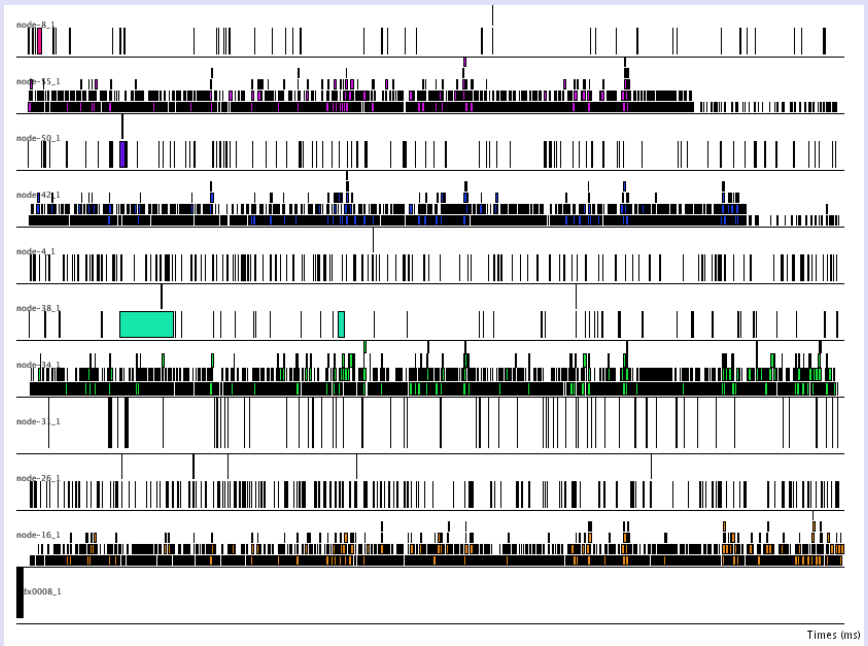
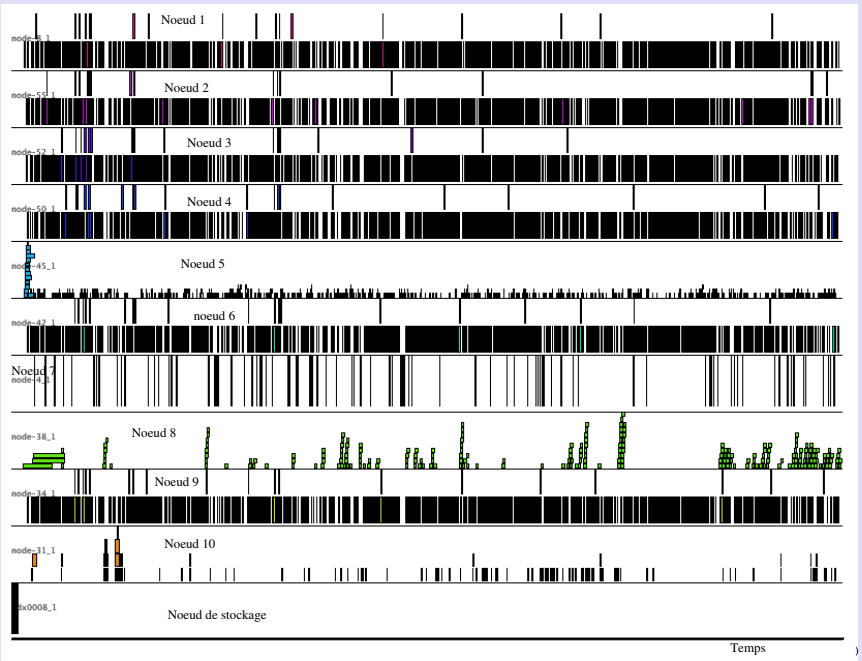


Diagramme de Gantt

Greedy



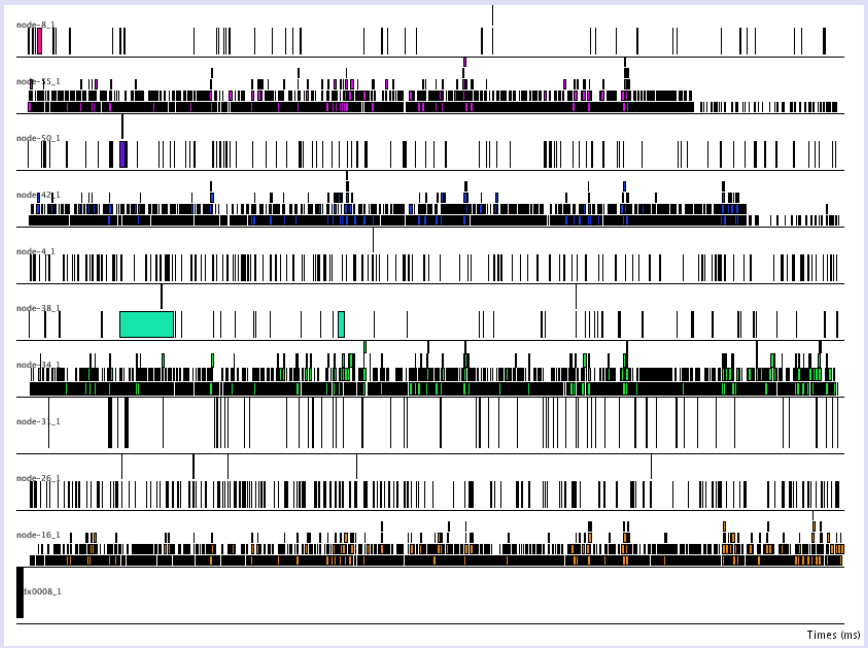
SRA



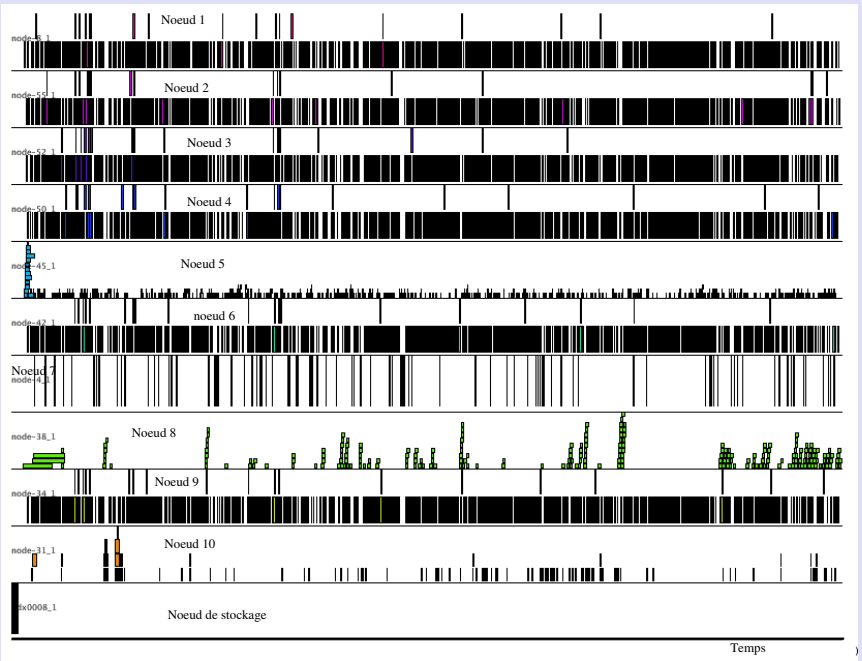
	glouton	SRA
temps de calcul	47953 s	41618 s

Diagramme de Gantt

Greedy



SRA



	glouton	SRA
temps de calcul	47953 s	41618 s

Plan de l'expos

Limites de la solution statique

- si les schémas ne sont pas suivis
- utilisation ponctuelle d'une banque

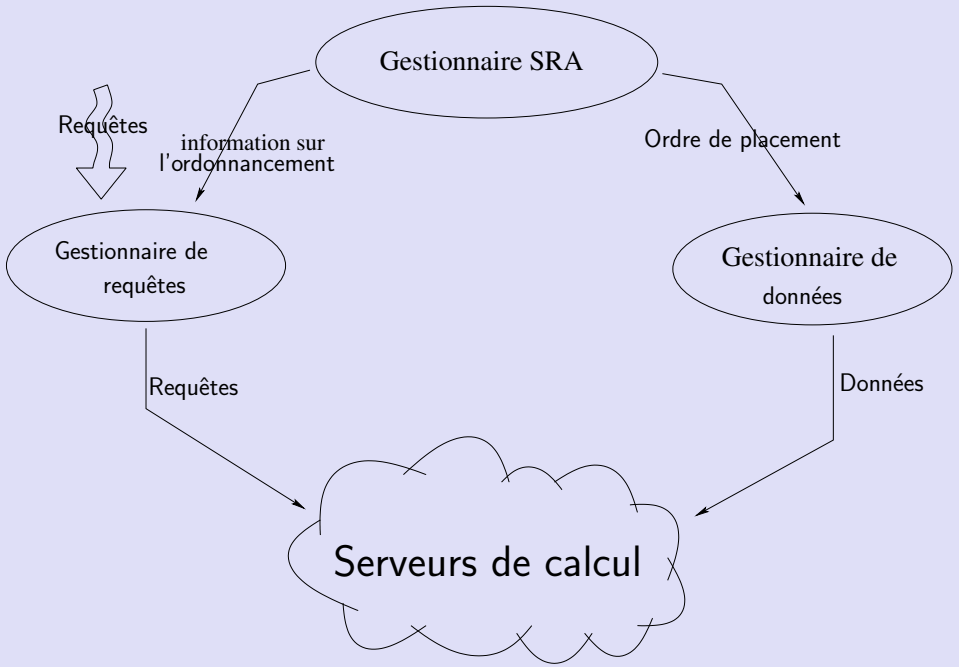
Limites de la solution statique

- si les schémas ne sont pas suivis
- utilisation ponctuelle d'une banque

Faire une version dynamique

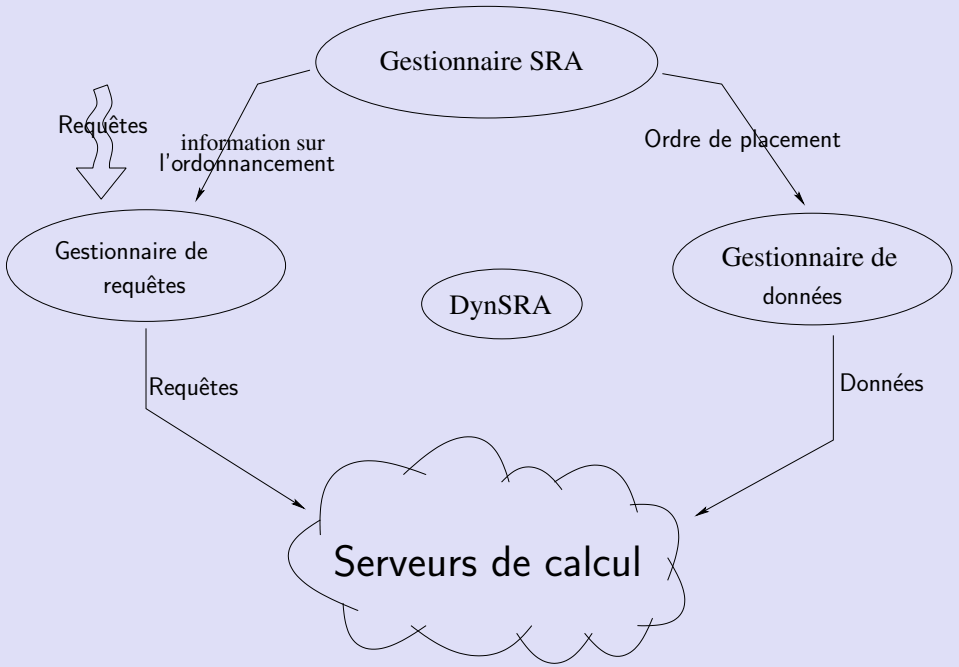
Détecter et décider

Architecture fonctionnelle



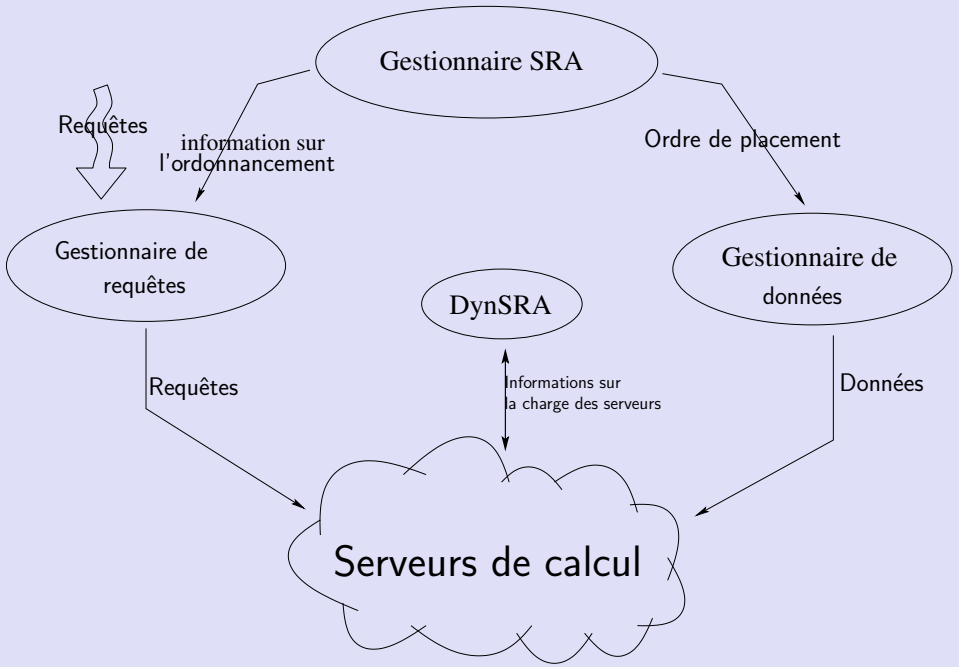
Détecter et décider

Architecture fonctionnelle



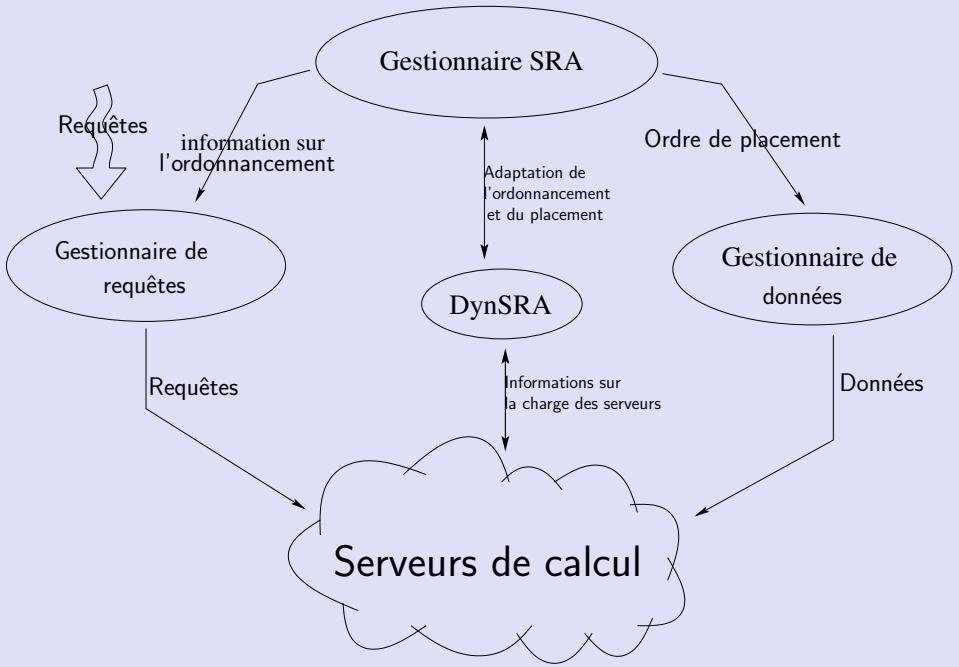
Détecter et décider

Architecture fonctionnelle



Détecter et décider

Architecture fonctionnelle



Algorithme général

1: **Tant que** il y a des requêtes à traiter **:**
2: Regarder la charge des serveurs
3: **Si** il y a des serveurs en surcharge **Alors**
4: Choisir le serveur $P_{surcharge}$ le plus surchargé
5: Déterminer la donnée D_{charge} cause de la surcharge
6: Choisir un serveur $P_{sous-charge}$ en sous-charge
7: **Si** il y a assez de place pour répliquer D_{charge} sur $P_{sous-charge}$ **Alors**
8: Répliquer
9: **Sinon**
10: **Tant que** il n'y a pas assez de place sur $P_{sous-charge}$ pour D_{charge} **:**
11: Choisir une donnée à supprimer sur D_{charge}
12: **Si** il n'y a toujours pas assez de place **Alors**
13: Restaurer les données supprimées
14: Choisir un autre serveur pour $P_{sous-charge}$ et recommencer à partir de 7
15: **Si** des données ont été répliquées ou supprimées **Alors**
16: Mettre à jour l'ordonnancement
17: Resoumettre les requêtes concernées
18: Attendre un temps $delai$

Algorithme général

1: **Tant que** il y a des requêtes à traiter **:**
2: Regarder la charge des serveurs
3: **Si** il y a des serveurs en surcharge **Alors**
4: Choisir le serveur $P_{surcharge}$ le plus surchargé
5: Déterminer la donnée D_{charge} cause de la surcharge
6: Choisir un serveur $P_{sous-charge}$ en sous-charge
7: **Si** il y a assez de place pour répliquer D_{charge} sur $P_{sous-charge}$ **Alors**
8: Répliquer
9: **Sinon**
10: **Tant que** il n'y a pas assez de place sur $P_{sous-charge}$ pour D_{charge} **:**
11: Choisir une donnée à supprimer sur D_{charge}
12: **Si** il n'y a toujours pas assez de place **Alors**
13: Restaurer les données supprimées
14: Choisir un autre serveur pour $P_{sous-charge}$ et recommencer à partir de 7
15: **Si** des données ont été répliquées ou supprimées **Alors**
16: Mettre à jour l'ordonnancement
17: Resoumettre les requêtes concernées
18: Attendre un temps $delai$

Algorithme général

1: **Tant que** il y a des requêtes à traiter **:**
2: Regarder la charge des serveurs
3: **Si** il y a des serveurs en surcharge **Alors**
4: Choisir le serveur $P_{surcharge}$ le plus surchargé
5: Déterminer la donnée D_{charge} cause de la surcharge
6: Choisir un serveur $P_{sous-charge}$ en sous-charge
7: **Si** il y a assez de place pour répliquer D_{charge} sur $P_{sous-charge}$ **Alors**
8: Répliquer
9: **Sinon**
10: **Tant que** il n'y a pas assez de place sur $P_{sous-charge}$ pour D_{charge} **:**
11: Choisir une donnée à supprimer sur D_{charge}
12: **Si** il n'y a toujours pas assez de place **Alors**
13: Restaurer les données supprimées
14: Choisir un autre serveur pour $P_{sous-charge}$ et recommencer à partir de 7
15: **Si** des données ont été répliquées ou supprimées **Alors**
16: Mettre à jour l'ordonnancement
17: Resoumettre les requêtes concernées
18: Attendre un temps $delai$

Algorithme général

1: **Tant que** il y a des requêtes à traiter **:**
2: Regarder la charge des serveurs
3: **Si** il y a des serveurs en surcharge **Alors**
4: Choisir le serveur $P_{surcharge}$ le plus surchargé
5: Déterminer la donnée D_{charge} cause de la surcharge
6: Choisir un serveur $P_{sous-charge}$ en sous-charge
7: **Si** il y a assez de place pour répliquer D_{charge} sur $P_{sous-charge}$ **Alors**
8: Répliquer
9: **Sinon**
10: **Tant que** il n'y a pas assez de place sur $P_{sous-charge}$ pour D_{charge} **:**
11: Choisir une donnée à supprimer sur D_{charge}
12: **Si** il n'y a toujours pas assez de place **Alors**
13: Restaurer les données supprimées
14: Choisir un autre serveur pour $P_{sous-charge}$ et recommencer à partir de 7
15: **Si** des données ont été répliquées ou supprimées **Alors**
16: Mettre à jour l'ordonnancement
17: Resoumettre les requêtes concernées
18: Attendre un temps $delai$

Algorithme général

1: **Tant que** il y a des requêtes à traiter **:**
2: Regarder la charge des serveurs
3: **Si** il y a des serveurs en surcharge **Alors**
4: Choisir le serveur $P_{surcharge}$ le plus surchargé
5: Déterminer la donnée D_{charge} cause de la surcharge
6: Choisir un serveur $P_{sous-charge}$ en sous-charge
7: **Si** il y a assez de place pour répliquer D_{charge} sur $P_{sous-charge}$ **Alors**
8: Répliquer
9: **Sinon**
10: **Tant que** il n'y a pas assez de place sur $P_{sous-charge}$ pour D_{charge} **:**
11: Choisir une donnée à supprimer sur D_{charge}
12: **Si** il n'y a toujours pas assez de place **Alors**
13: Restaurer les données supprimées
14: Choisir un autre serveur pour $P_{sous-charge}$ et recommencer à partir de 7
15: **Si** des données ont été répliquées ou supprimées **Alors**
16: Mettre à jour l'ordonnancement
17: Resoumettre les requêtes concernées
18: Attendre un temps $delai$

Algorithme général

1: **Tant que** il y a des requêtes à traiter **:**
2: Regarder la charge des serveurs
3: **Si** il y a des serveurs en surcharge **Alors**
4: Choisir le serveur $P_{surcharge}$ le plus surchargé
5: Déterminer la donnée D_{charge} cause de la surcharge
6: Choisir un serveur $P_{sous-charge}$ en sous-charge
7: **Si** il y a assez de place pour répliquer D_{charge} sur $P_{sous-charge}$ **Alors**
8: Répliquer
9: **Sinon**
10: **Tant que** il n'y a pas assez de place sur $P_{sous-charge}$ pour D_{charge} **:**
11: Choisir une donnée à supprimer sur D_{charge}
12: **Si** il n'y a toujours pas assez de place **Alors**
13: Restaurer les données supprimées
14: Choisir un autre serveur pour $P_{sous-charge}$ et recommencer à partir de 7
15: **Si** des données ont été répliquées ou supprimées **Alors**
16: Mettre à jour l'ordonnancement
17: Resoumettre les requêtes concernées
18: Attendre un temps $delai$

Heuristiques - 1/3

Choix du serveur en surcharge

- MaxCompServ
  - plus grand temps de calcul en attente
- MaxOverMedianServ
  - plus grand écart en volume de calcul entre temps de calcul et temps de calcul moyen

Cause de la surcharge

- MaxCompData
  - donnée représentant le plus grand volume de calcul
- MaxDiffData
  - donnée dont la différence entre utilisation prévue et réelle est la plus grande

Heuristiques - 1/3

Choix du serveur en surcharge

- MaxCompServ
  - plus grand temps de calcul en attente
- MaxOverMedianServ
  - plus grand écart en volume de calcul entre temps de calcul et temps de calcul moyen

Cause de la surcharge

- MaxCompData
  - donnée représentant le plus grand volume de calcul
- MaxDiffData
  - donnée dont la différence entre utilisation prévue et réelle est la plus grande

Heuristiques - 2/3

Choix du serveur cible

- MinTimeServ
  - plus court temps de calcul en attente
- MaxUnderMedianServ
  - plus grand écart en volume de calcul entre temps de calcul et temps de calcul moyen

Suppression des données

- MinCompData
  - donnée représentant le plus petit volume de calcul
- LessUsedData
  - donnée dont la fréquence d'utilisation est la plus faible

Heuristiques - 2/3

Choix du serveur cible

- MinTimeServ
  - plus court temps de calcul en attente
- MaxUnderMedianServ
  - plus grand écart en volume de calcul entre temps de calcul et temps de calcul moyen

Suppression des données

- MinCompData
  - donnée représentant le plus petit volume de calcul
- LessUsedData
  - donnée dont la fréquence d'utilisation est la plus faible

Heuristiques - 3/3

Réordonnancer

- LPSched
  - recalcul du programme linéaire
- EqShareSched
  - partage égal entre le nouveau et l'ancien serveur
- MedianCompSched
  - on redistribue les calculs pour que l'un des 2 serveurs revienne a la moyenne

Environnement

Simulateur

version modifiée d'Optorsim

Plate-formes

générées par Tiers

Requêtes

Ensemble de 40000 requêtes :

- généré à partir des jeux de requêtes réelles
- différent de la distribution originale
- 3 types
  - tous : tous les types changent
  - min10 : juste les 10 + fréquentes

Environnement

Simulateur

version modifiée d'Optorsim

Plate-formes

générées par Tiers

Requêtes

Ensemble de 40000 requêtes :

- généré à partir des jeux de requêtes réelles
- différent de la distribution originale
- 3 types
  - tous : tous les types changent
  - min10 : juste les 10 - fréquentes
  - max10 : juste les 10 + fréquentes

Environnement

Simulateur

version modifiée d'Optorsim

Plate-formes

générées par Tiers

Requêtes

Ensemble de 40000 requêtes :

- généré à partir des jeux de requêtes réelles
- différent de la distribution originale
- 3 types
  - tous : tous les types changent
  - min10 : juste les 10 - fréquentes
  - max10 : juste les 10 + fréquentes

Environnement

Simulateur

version modifiée d'Optorsim

Plate-formes

générées par Tiers

Requêtes

Ensemble de 40000 requêtes :

- généré à partir des jeux de requêtes réelles
- différent de la distribution originale
- 3 types
  - tous : tous les types changent
  - min10 : juste les 10 - fréquentes
  - max10 : juste les 10 + fréquentes

Environnement

Simulateur

version modifiée d'Optorsim

Plate-formes

générées par Tiers

Requêtes

Ensemble de 40000 requêtes :

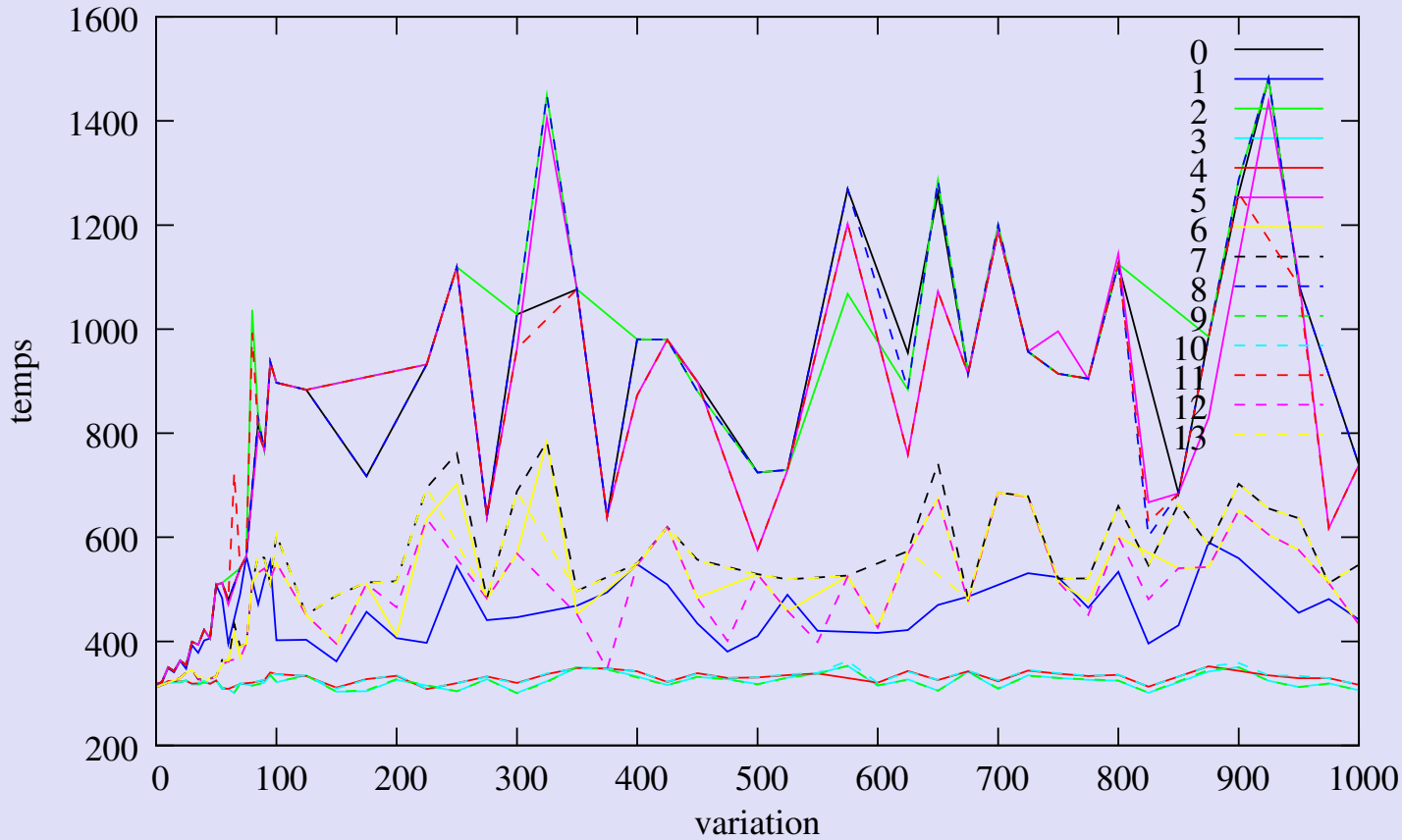
- généré à partir des jeux de requêtes réelles
- différent de la distribution originale
- 3 types
  - tous : tous les types changent
  - min10 : juste les 10 - fréquentes
  - max10 : juste les 10 + fréquentes

Heuristiques testées

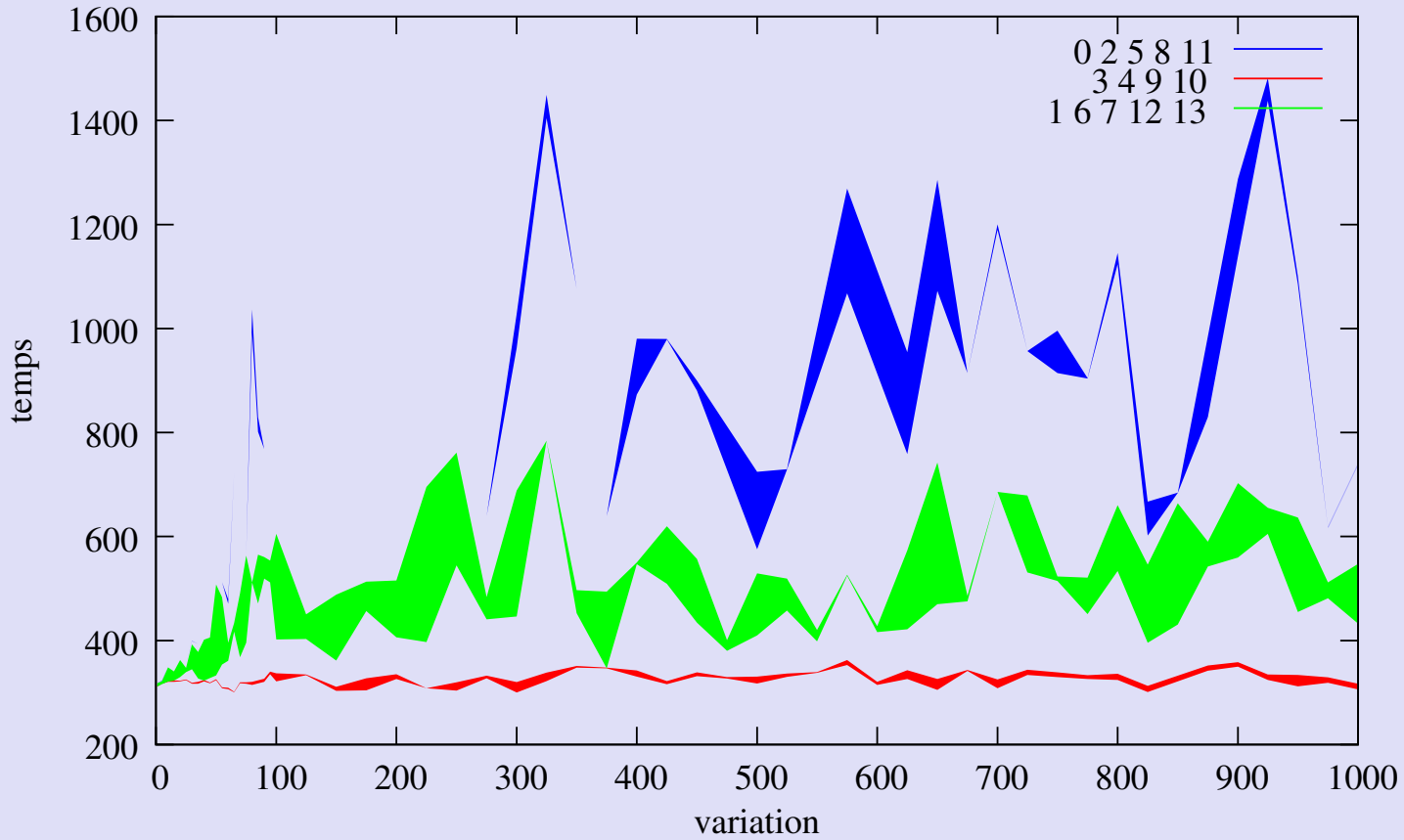
13 combinaisons

	serveur surchargé	données de surcharge	serveur sous-chargé	données à supprimer	ordonnancement
0	aucun	aucun	aucun	aucun	aucun
1	MaxTimeServ	MaxCompData	MinTimeServ	aucun	EqShareSched
2	MaxTimeServ	MaxCompData	MinTimeServ	MinCompData	LPSched
3	MaxTimeServ	MaxCompData	MinTimeServ	MinCompData	EqShareSched
4	MaxTimeServ	MaxCompData	MinTimeServ	MinCompData	MedianCompSched
5	MaxTimeServ	MaxDiff	MinTimeServ	LessUsed	LPSched
6	MaxTimeServ	MaxDiff	MinTimeServ	LessUsed	EqShare
7	MaxTimeServ	MaxDiff	MinTimeServ	LessUsed	MedianCompSched
8	MaxOverMedianServ	MaxCompData	MinOverMedianServ	MinCompData	LPSched
9	MaxOverMedianServ	MaxCompData	MinOverMedianServ	MinCompData	EqShare
10	MaxOverMedianServ	MaxCompData	MinOverMedianServ	MinCompData	MedianCompSched
11	MaxOverMedianServ	MaxDiff	MinOverMedianServ	LessUsed	LPSched
12	MaxOverMedianServ	MaxDiff	MinOverMedianServ	LessUsed	EqShare
13	MaxOverMedianServ	MaxDiff	MinOverMedianServ	LessUsed	MedianCompSched

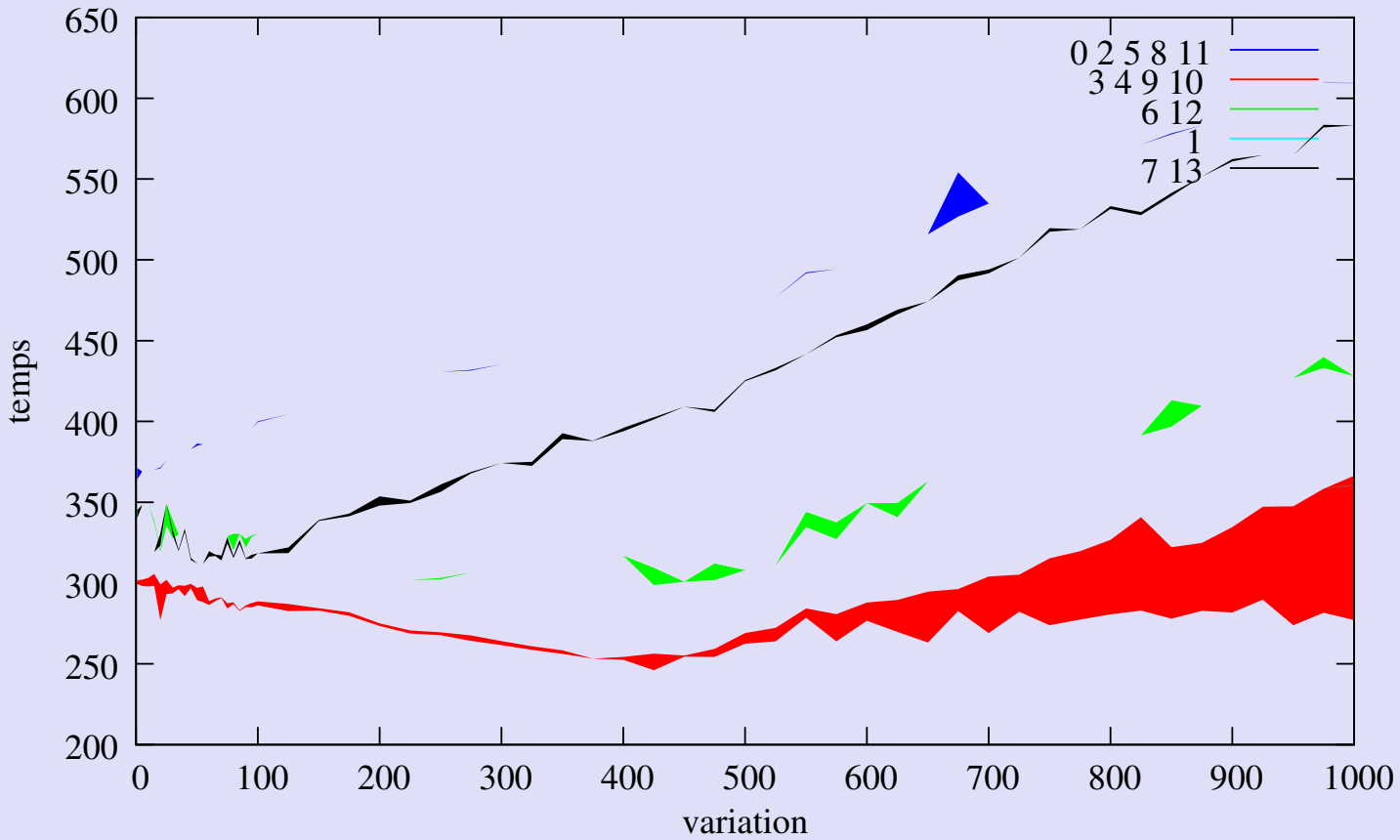
Temps d'exécution : tous différents



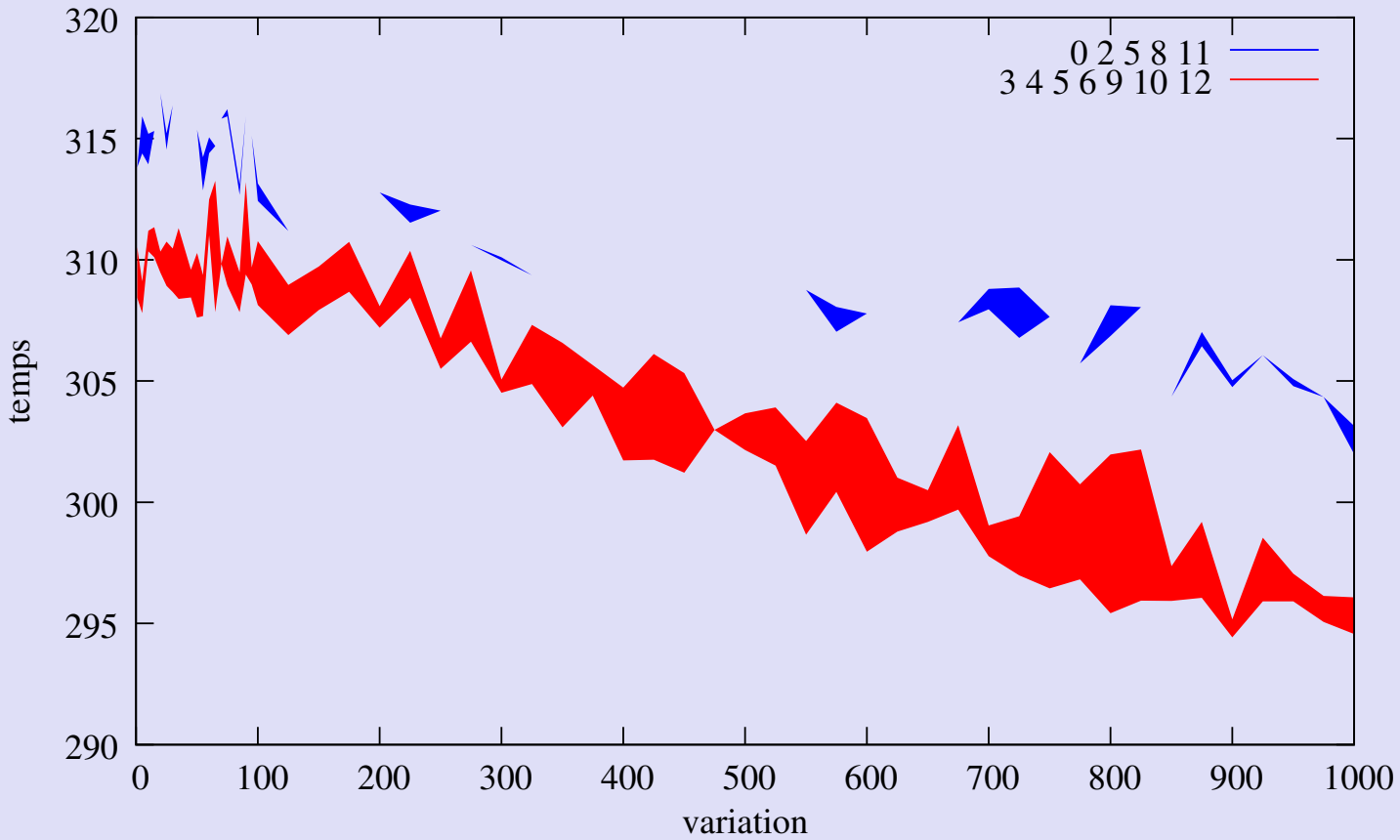
Temps d'exécution : tous différents



Temps d'exécution : min10



Temps d'exécution : max10



Heuristiques testées

13 combinaisons

	serveur surchargé	données de surcharge	serveur sous-chargé	données à supprimer	ordonnancement
0	aucun	aucun	aucun	aucun	aucun
1	MaxTimeServ	MaxCompData	MinTimeServ	aucun	EqShareSched
2	MaxTimeServ	MaxCompData	MinTimeServ	MinCompData	LPSched
3	MaxTimeServ	MaxCompData	MinTimeServ	MinCompData	EqShareSched
4	MaxTimeServ	MaxCompData	MinTimeServ	MinCompData	MedianCompSched
5	MaxTimeServ	MaxDiff	MinTimeServ	LessUsed	LPSched
6	MaxTimeServ	MaxDiff	MinTimeServ	LessUsed	EqShare
7	MaxTimeServ	MaxDiff	MinTimeServ	LessUsed	MedianCompSched
8	MaxOverMedianServ	MaxCompData	MinOverMedianServ	MinCompData	LPSched
9	MaxOverMedianServ	MaxCompData	MinOverMedianServ	MinCompData	EqShareSched
10	MaxOverMedianServ	MaxCompData	MinOverMedianServ	MinCompData	MedianCompSched
11	MaxOverMedianServ	MaxDiff	MinOverMedianServ	LessUsed	LPSched
12	MaxOverMedianServ	MaxDiff	MinOverMedianServ	LessUsed	EqShare
13	MaxOverMedianServ	MaxDiff	MinOverMedianServ	LessUsed	MedianCompSched

Heuristiques testées

13 combinaisons

	serveur surchargé	données de surcharge	serveur sous-chargé	données à supprimer	ordonnancement
0	aucun	aucun	aucun	aucun	aucun
1	MaxTimeServ	MaxCompData	MinTimeServ	aucun	EqShareSched
2	MaxTimeServ	MaxCompData	MinTimeServ	MinCompData	LPSched
3	MaxTimeServ	MaxCompData	MinTimeServ	MinCompData	EqShareSched
4	MaxTimeServ	MaxCompData	MinTimeServ	MinCompData	MedianCompSched
5	MaxTimeServ	MaxDiff	MinTimeServ	LessUsed	LPSched
6	MaxTimeServ	MaxDiff	MinTimeServ	LessUsed	EqShare
7	MaxTimeServ	MaxDiff	MinTimeServ	LessUsed	MedianCompSched
8	MaxOverMedianServ	MaxCompData	MinOverMedianServ	MinCompData	LPSched
9	MaxOverMedianServ	MaxCompData	MinOverMedianServ	MinCompData	EqShareSched
10	MaxOverMedianServ	MaxCompData	MinOverMedianServ	MinCompData	MedianCompSched
11	MaxOverMedianServ	MaxDiff	MinOverMedianServ	LessUsed	LPSched
12	MaxOverMedianServ	MaxDiff	MinOverMedianServ	LessUsed	EqShare
13	MaxOverMedianServ	MaxDiff	MinOverMedianServ	LessUsed	MedianCompSched

Heuristiques testées

13 combinaisons

	serveur surchargé	données de surcharge	serveur sous-chargé	données à supprimer	ordonnancement
0	aucun	aucun	aucun	aucun	aucun
1	MaxTimeServ	MaxCompData	MinTimeServ	aucun	EqShareSched
2	MaxTimeServ	MaxCompData	MinTimeServ	MinCompData	LPSched
3	MaxTimeServ	MaxCompData	MinTimeServ	MinCompData	EqShareSched
4	MaxTimeServ	MaxCompData	MinTimeServ	MinCompData	MedianCompSched
5	MaxTimeServ	MaxDiff	MinTimeServ	LessUsed	LPSched
6	MaxTimeServ	MaxDiff	MinTimeServ	LessUsed	EqShare
7	MaxTimeServ	MaxDiff	MinTimeServ	LessUsed	MedianCompSched
8	MaxOverMedianServ	MaxCompData	MinOverMedianServ	MinCompData	LPSched
9	MaxOverMedianServ	MaxCompData	MinOverMedianServ	MinCompData	EqShareSched
10	MaxOverMedianServ	MaxCompData	MinOverMedianServ	MinCompData	MedianCompSched
11	MaxOverMedianServ	MaxDiff	MinOverMedianServ	LessUsed	LPSched
12	MaxOverMedianServ	MaxDiff	MinOverMedianServ	LessUsed	EqShare
13	MaxOverMedianServ	MaxDiff	MinOverMedianServ	LessUsed	MedianCompSched

Plan de l'expos

Algorithmes d'ordonnancement et de réplication

Statique

- modélisation en régime permanent
- validé par simulation et expérimentalement
  - simulateur basé sur Optorsim
  - prototype dans DIET/DTM
- bonne optimisation pour
  - réseau lent
  - espace de stockage limité

Dynamique

- nombreuses heuristiques
- bons résultats pour certaines
- stabilise le temps d'exécution

Futur proche

Implémentation de DynSRA

- DIET/DTM
- CORI pour la remontée d'informations

Prise en compte des coûts de transfert

- thèse de Gaël Lemahec (CC IN2P3 / LIP)

À long terme

Placement et ordonnancement conjoint

- domaine récent
- problème NP-complet
  - le problème du placement à lui seul est NP-complet
- accroissement de la taille des données
  - bioinformatique, physique, océanographie, ...
- d'autres modèles d'applications

À long terme

Placement et ordonnancement conjoint

- domaine récent
- problème NP-complet
  - le problème du placement à lui seul est NP-complet
- accroissement de la taille des données
  - bioinformatique, physique, océanographie, ...
- d'autres modèles d'applications

Le sujet reste très ouvert