Intégration de l'Ordonnancement et de la Réplication dans les Data Grids

Turki Tarek
tartur2000@yahoo.fr
Université Claude Bernard Lyon

Présentation de l'article :

« Integration of Scheduling and Replication in Data Grids » par Chakrabarti et al.

Plan

1. Introduction
2. Objectifs et Hypothèses
3. Positionnement du problème
   i. Ordonnancement des Jobs
   ii. Réplication des données
4. Présentation de l'approche IRS
   i. Stratégies d'Ordonnancement des Jobs
   ii. Stratégie de Réplication de Données
5. Évaluation et conclusion

Introduction

- ***Objectifs de la Data Grid:*** partager des données volumineuses (10^{15} octets) sur une infrastructure réseau
- ***Domaines d'application:*** Physique nucléaire, Biologie, Imagerie médicale, …
- ***Middleware de la Data Grid*:**
  - Workload Scheduling and Management
  - Data Management
  - Grid Monitoring services
  - Mass Storage Management
  - Local Fabric management

Objectif principal

- La latence d'un job dépend de la ressource de calcul et de l'emplacement des données requises
- Pour réduire cette latence il faut :
  - Diminuer la latence liée au calcul et à l'attente en ordonnançant les jobs
  - Diminuer la latence liée aux données en les répliquant

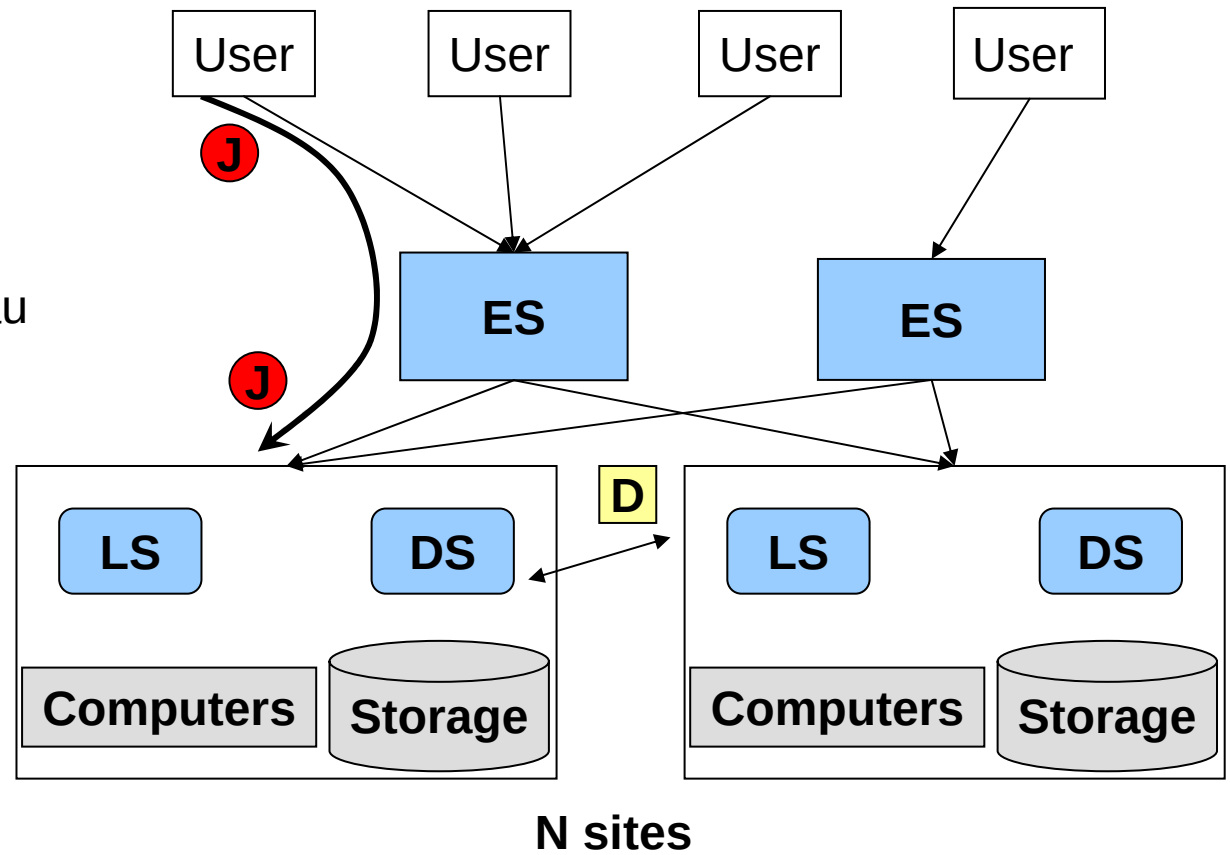
⇒ Combiner les stratégies d'ordonnancement et de Réplication pour un meilleur résultat [Ranganathan]

Modèle proposé par Ranganathan

- La grille est composée d’
  - Un ensemble de sites
  - Un ensemble de données
  - Un ensemble d’Utilisateurs soumettant des Jobs
- Un site est composé de 3 parties
  - Un Ordonnanceur externe (**ES**)
  - Un Ordonnanceur local (**LS**)
  - Un Ordonnanceur de données (**DS**)

Architecture du modèle

1. **User** soumet un Job **J** à **ES**
2. **ES** envoie **J** au site approprié
3. **LS** ordonnance les jobs au sain d'un site
4. **LS** demande à **DS** la donnée requise
5. **DS** s'occupe de la réplication et le mouvement des données



Limites

- Un job ne requiert qu'une seule donnée pour son exécution
- Les stratégies d'ordonnancement qui en découlent sont simples
- La stratégie de réplication n'influe pas vraiment sur le résultat

Plan

1. Introduction
2. Objectifs et Hypothèses
3. Positionnement du problème
   i. Ordonnancement des Jobs
   ii. Réplication des données
4. Présentation de l'approche IRS
   i. Stratégies d'Ordonnancement des Jobs
   ii. Stratégie de Réplication de Données
5. Évaluation et conclusion

Objectifs

- Une stratégie de réplication des données qui se base sur l'historique du trafic

- Une stratégie d'ordonnancement des jobs qui minimise la latence globale (entre la soumission d'un job et son exécution)

- Évaluer les algorithmes à travers des simulations dans un contexte général

Hypothèses

1. La grille est un graphe non orienté.
2. On utilise l'architecture de Ranganathan
3. On utilise un **ES** centralisé
4. Chaque site possède un **LS** et des politiques d'ordonnancement propres
5. La grille est plus ou moins stable
6. Les données sont assimilées à des fichiers

Hypothèses (suite)

7. La donnée est accédée en lecture seule
8. Les jobs ne sont pas préemptifs et leurs temps d'exécution dépendent des tailles des fichiers requis.
9. Les fichiers sont transmis séquentiellement
10. L'historique du trafic est donnée pour un intervalle de temps **T**. Cette information est stockée dans un emplacement central

Plan

1. Introduction
2. Objectifs et Hypothèses
3. Positionnement du problème
   i. Ordonnancement des Jobs
   ii. Réplication des données
4. Présentation de l'approche IRS
   i. Stratégies d'Ordonnancement des Jobs
   ii. Stratégie de Réplication de Données
5. Évaluation et conclusion

Modélisation

- On modélise un job **J** par $J = \langle S, \tilde{F}, \tilde{C} \rangle$

  S est le site où **J** est exécuté, **F̃** est la liste des fichiers requis par **J** et **C̃** est le temps de calcul de **J**

- Un site **S** est modélisé par $S = \langle \hat{F}, V, P_s \rangle$

  F est la l'ensemble des fichiers stockés par **S**, **V** est sa capacité de stockage et **Ps** est sa capacité de calcul.

Ordonnancement des Jobs (JS)

- Soit $\mathbf{J_i}$ un job, et $\hat{S} = \{S_1, S_2, \ldots, S_n\}$ l'ensemble des sites de la grille

- À quel site $\mathbf{S_j}$ faut-il affecter $\mathbf{J_i}$ pour minimiser sa latence d'exécution?

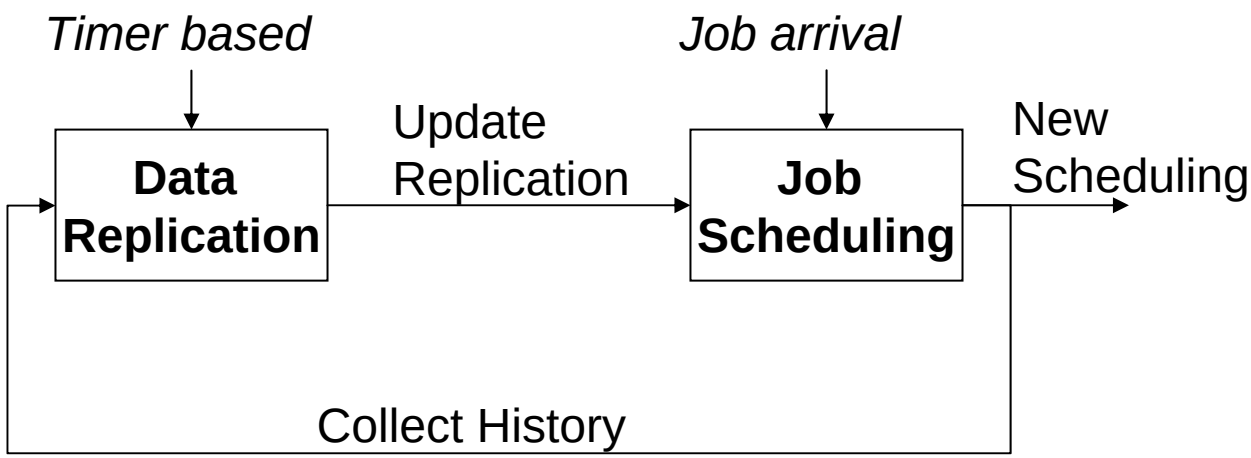
Réplication des données (DR)

- Soit $D^{T}_{F_i S_j}$ la matrice de demande créée à partir de l'historique du trafic des données pendant la période **T**. **S** l'ensemble des sites

- Comment distribuer l'ensemble des fichiers sur **S** pour minimiser la latence, en tenant compte de $D^{T}_{F_i S_j}$ et de la capacité de stockage de chaque site?

Solution proposée

- *Latence* = $latence_{calcul}$ + $latence_{data}$ + $latence_{queue}$
- L'approche **IRS** (*Integrated Replication and Scheduling*) proposée est une combinaison des stratégies de réplication et d'ordonnancement



Plan

1. Introduction
2. Objectifs et Hypothèses
3. Positionnement du problème
   i. Ordonnancement des Jobs
   ii. Réplication des données
4. Présentation de l'approche IRS
   i. Stratégies d'Ordonnancement des Jobs
   ii. Stratégie de Réplication de Données
5. Évaluation et conclusion

Définitions

- ***Taux de demande*** d'un fichier $\mathbf{F_i}$ $\eta_{F_i} = \dfrac{\sum_{j=1}^{n} D_{F_i S_j}}{\sum_{j=1}^{n}\sum_{i=1}^{m} D_{F_i S_j}}$
- ***Latence d'un fichier*** $\mathbf{F_k}$ Δ_{ij}^{k} : est le temps de transfert du fichier de $\mathbf{S_i}$ à $\mathbf{S_j}$
- ***Latence de calcul*** d'un job i dans un site $\mathbf{S_j}$

$$\omega_{ij} = \frac{\sum_{i=1}^{m} \tau_i}{P_j}$$

$\langle \tau_1, \tau_2, \ldots \tau_m \rangle$ *sont les tailles des fichiers requis par J*

Définitions (suite)

- ***Latence d'attente*** d'un job i dans le site S_j

$$Q_{ij} = q_j \frac{\sum_{i=1}^{m} \tau_i}{P_j}$$, q_j est la taille de la file d'attente à S_j

- ***Le nombre de slots disponibles*** γ_j représente le nombre moyen de fichiers que le site S_j peut stocker

$$\gamma_j = \frac{V_j}{\bar{\tau}} \qquad \bar{\tau} \text{ est la taille moyenne d'un fichier}$$

Stratégies d'ordonnancement de jobs

L'algorithme **JS** est constitué de deux parties :

- Stratégie d'ordonnancement, 2 stratégies ont été proposées
  - Ordonnancement par association (**MJS**)
  - Ordonnancement basé sur le coût (**CJS**)

- Mise à Jour de la matrice de demande $D^{T}_{F_i S_j}$

Ordonnancement par association (MJS)

- **Principe :** affecter le job au site possédant le plus grand nombre de fichiers requis.
- **MJS** se base sur le facteur **ν,** dont la formule est donnée ci-dessous, pour distribuer les jobs. m est le nombre maximum de fichiers associés au job dans le site, q est la taille moyenne des files d'attente, q_i est la taille de la file d'attente du site et α est généralement égal à 1

$$\nu = m \cdot \alpha \cdot \frac{\bar{q}}{q_i}$$

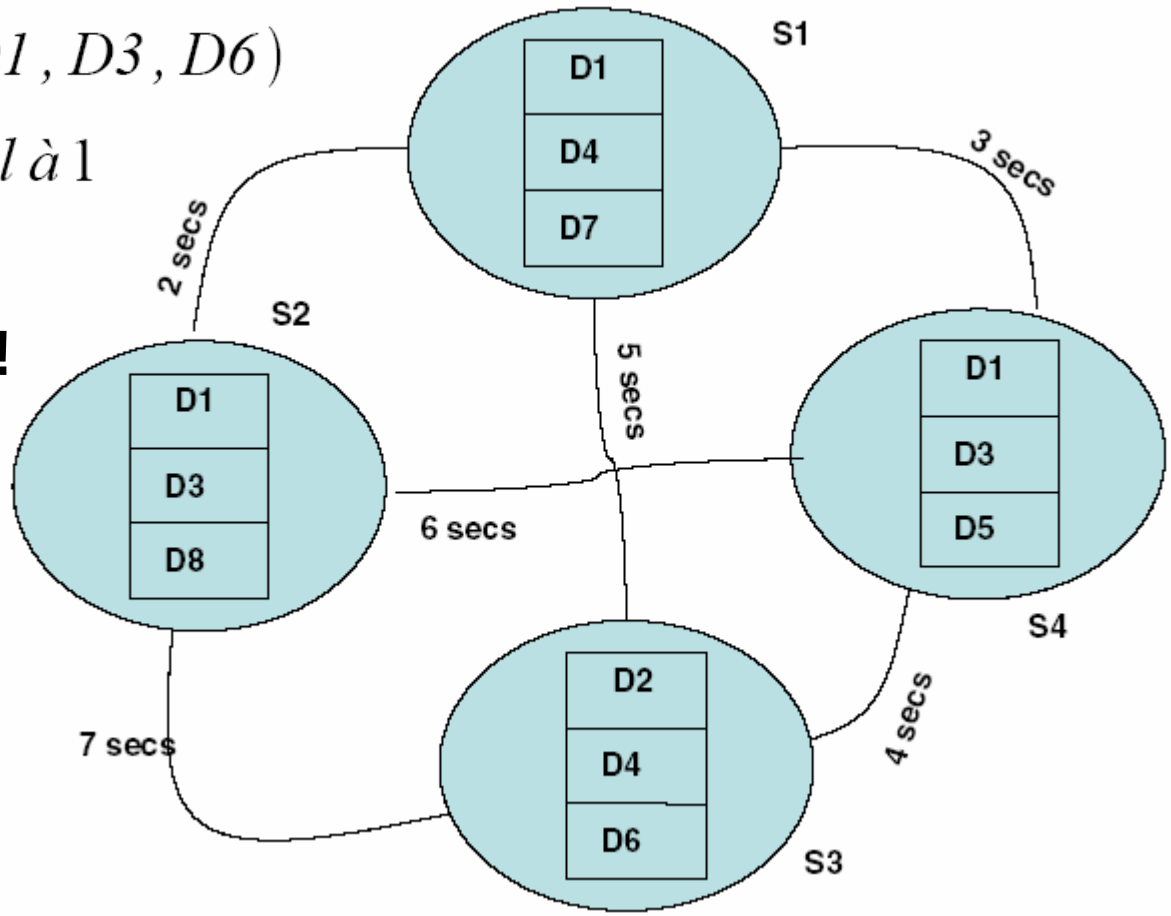
- Grâce à **ν**, on évite la possibilité d'envoyer les jobs toujours au même site.

Exemple

$Soit\ J_0\ tel\ que\ \tilde{F}_0 = (D1, D3, D6)$

$\alpha \cdot \frac{\bar{q}}{q_i}\ considéré\ est\ égal\ à\ 1$

J_0 est affecté à **S4!**



Ordonnancement basé sur le coût (CJS)

- Soit C^s_{ij} le coût d'ordonnancement d'un job J_i dans le site S_j.

- C^s_{ij} contient la latence liée au déplacement des données, celle liée au calcul du job et le temps d'attente dans le site

- Le job est affecté au site qui minimise cette fonction de coût.

Ordonnancement avec prise en compte de l'ordre des données

- Au départ le job peut ne pas avoir besoin de la totalité des données requises.
- Exemple: J nécessite (f1,f2,f3) au départ et (f4,f5) plus tard, la formule de la latence devient:

$$Latency = max\left(\sum_{s=1}^{3}\Delta_{ij}^{s}+\sum_{s=1}^{3}\omega_{ij}, \sum_{s=4}^{5}\Delta_{ij}^{s}\right)+\sum_{s=4}^{5}\omega_{ij}$$

- En généralisant cette expression on obtient la latence pour chaque étape i:

$$L(i) = \max(L(i-1), \sum_{j=1}^{K_i}\Delta_{ij}) + \sum_{j=1}^{K_i}\omega_{ij}$$

$$L(1) = \sum_{j=1}^{K_1}\Delta^{s}_{ij} + \sum_{j=1}^{K_1}\varpi_{ij}$$

Plan

1. Introduction
2. Objectifs et Hypothèses
3. Positionnement du problème
   i. Ordonnancement des Jobs
   ii. Réplication des données
4. Présentation de l'approche IRS
   i. Stratégies d'Ordonnancement des Jobs
   ii. Stratégie de Réplication de Données
5. Évaluation et conclusion

Stratégie de Réplication de donnée (DRS)

- La stratégie de réplication des données se décompose en 2 étapes:
  - Allocation de la limite de réplication pour chaque fichier
    - La limite de réplication χ_i d'un fichier F_i est définie comme étant le nombre de sites où F_i doit être répliqué
    - La formule donnant est la suivante

$$\chi_i = \mathit{min}\left(|\mathit{sites}|, 1 + \left(\mathit{ceiling}\left(\eta_{F_i} \cdot \left(\sum_{j=1}^{n} \gamma_j - \Phi\right)\right)\right)\right)$$

  - La deuxième étape est la Réplication

Réplication de la donnée

- Soit δ_{ij} la latence prévisionnelle pour un fichier F_i répliqué sur S_j et soit p_{ik} la probabilité pour qu'un job affecté à S_k nécessite F_i alors : $$\delta_{ij} = \sum_{k=1}^{n} p_{ik} \cdot l_{kj} \cdot \tau_i$$
- De la même façon la latence de calcul prévisionnelle est définie par : $$\theta_{ij} = \sum_{k=1}^{n} p_{ij} \cdot (\tau_j / P_j)$$
- Et aussi la latence d'attente prévisionnelle pour que F_i soit affecté à S_j : $$\alpha_{ij} = q_j \cdot \theta_{ij}$$

$$\Rightarrow \quad \boxed{C_{ij}^{R} = \delta_{ij} + \theta_{ij} + \alpha_{ij}}$$

Algorithme de Réplication

1. On commence la réplication par le fichier ayant le plus grand η_{Fi}
2. Ensuite on calcule la limite de réplication χ_i du fichier F_i (χ_i augmente quand η_{Fi} augmente)
3. Les meilleurs χ_i sites sont choisis parmi les n sites en minimisant la valeur du coût C^R_{ij}
4. On recommence le processus pour le fichier ayant le second plus grand χ_i et ainsi de suite, jusqu'à ce qu'il ne reste aucun fichier.

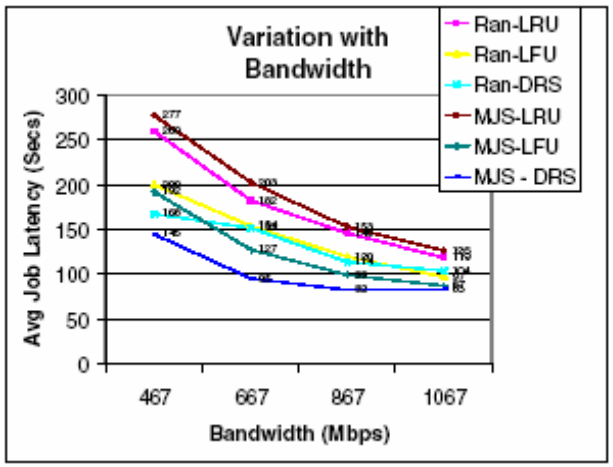
Plan

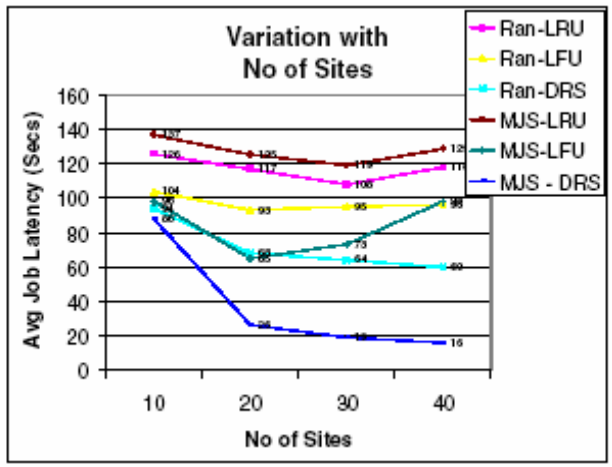
1. Introduction
2. Objectifs et Hypothèses
3. Positionnement du problème
   i. Ordonnancement des Jobs
   ii. Réplication des données
4. Présentation de l'approche IRS
   i. Stratégies d'Ordonnancement des Jobs
   ii. Stratégie de Réplication de Données
5. Évaluation et conclusion

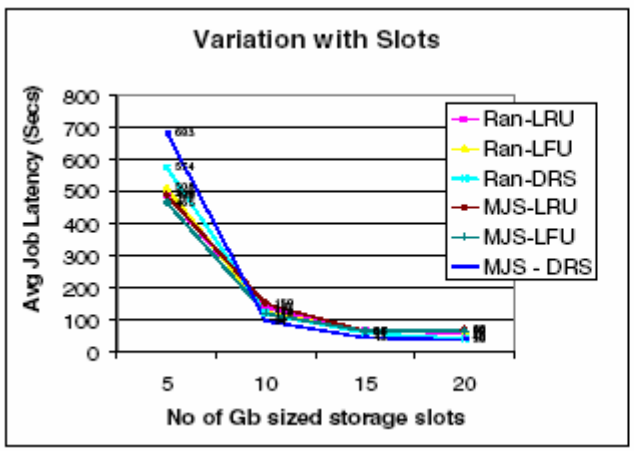
Évaluation des performances

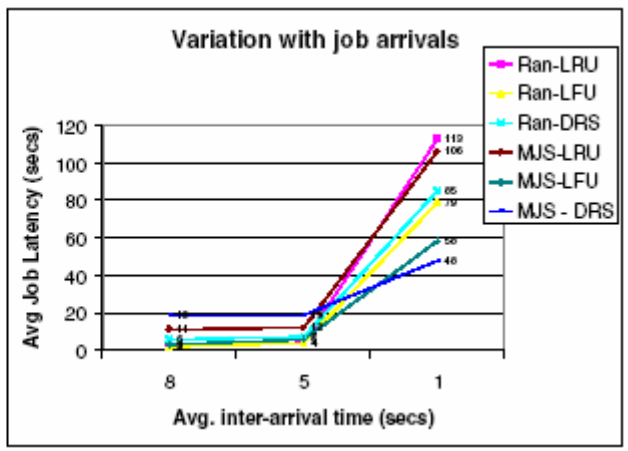
- La simulation est effectué à l'aide d'**OptorSim**
- Les deux stratégies proposées **MJS** et **DRS** sont comparées à **LRU** (*Least Recently Used*) et **LFU** (*Least Frequently Used*)
- Les résultats des approches n'utilisant pas de stratégie de réplication ne sont pas montrés. (latence de 10 à 12 fois plus importante)

Résultats des simulations









Conclusion

- L'approche **IRS** (*Integrated Replication and Scheduling*) combine les stratégies de Réplication et d'Ordonnancement
- Ordonnancement des jobs en **MJS** ou **CJS**
- Réplication des données dépend de l'historique
- L'approche **MJS-DRS** a montré des résultats prometteurs
- L'architecture utilisée est limitée par la taille de la grille (**ES** centralisé)

Bibliographie

- A. Chakrabarti, R.A. Dheepak et S. Sengupta, **« Integration of Scheduling and Replication in Data Grids »**, *Infosys* 2004
- K. Ranganathan et I. Foster**, « Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids »**, *in journal of grid computing*, vol. 1, no. 2, pp. 53-62, Apr 2003
- www.eu-datagrid.org

Any Questions ?