

Simultaneous scheduling of data replication and computation in Grids

Frédéric Desprez, Antoine Vernois

Laboratoire de l'Informatique du Parallélisme
École Normale Supérieure de Lyon, France

CLADE 2005
July, 24th 2005

Agenda

- 1 Motivations
- 2 Model
- 3 Simulations et results
- 4 Conclusions

Agenda

- 1 Motivations
 - Bioinformatic applications
 - Replication
- 2 Model
- 3 Simulations et results
- 4 Conclusions

Context : bioinformatic application

- reference databanks

- ▶ "flat" text files
- ▶ few MB to several GB
- ▶ update : daily to monthly
- ▶ number and size of data increase very quickly

- requests

- ▶ one algorithm applied to one or two databanks

Context : bioinformatic application

- reference databanks
 - ▶ "flat" text files
 - ▶ few MB to several GB
 - ▶ update : daily to monthly
 - ▶ number and size of data increase very quickly
- requests
 - ▶ one algorithm applied to one or two databanks

Context : bioinformatic application

- reference databanks
 - ▶ "flat" text files
 - ▶ few MB to several GB
 - ▶ update : daily to monthly
 - ▶ number and size of data increase very quickly
- requests
 - ▶ one algorithm applied to one or two databanks

Logs analysis

From log files of an existing bioinformatic portal :

- some requests are more frequent than other
 - ▶ blast over sp.fas : 77% of requests
- the usage of databanks and kind of requests is constant.

Logs analysis

From log files of an existing bioinformatic portal :

- some requests are more frequent than other
 - ▶ blast over sp.fas : 77% of requests
- the usage of databanks and kind of requests is constant.

Logs analysis

From log files of an existing bioinformatic portal :

- some requests are more frequent than other
 - ▶ blast over sp.fas : 77% of requests
- the usage of databanks and kind of requests is constant.

We can start from the study of previous usage schemes.

Some figures

extract from logs of NPS@, bioinformatic web portal :

Number of databanks	23
Number of algorithms	8
Number of couple algorithm-databanks	80
Size of the smallest databank	1 MB
Size of the largest databank	12 GB

Replicate databanks

Goal :

- improve computation time,
- and/or platform throughput.

Data sets are initially stored on public server :

- insert them into the grid
- keep them up to date
- prevent bottleneck

Replicate databanks

Goal :

- improve computation time,
- and/or platform throughput.

Data sets are initially stored on public server :

- insert them into the grid
- keep them up to date
- prevent bottleneck

Question : Where and when create replicas ??

A simple idea

Store all databanks on each server.

- not always possible : too many data
- too much space occupied by useless databanks
 - ▶ databanks are not all used at all the time. (embl.fas = 12 Gb, < 1% of requests)
- updating databanks become costly

Usually

- scheduling and replication are two independent processes,
- replication has to be done by users,
- schedulers don't take care of locality of data.

Usually

- scheduling and replication are two independent processes,
- replication has to be done by users,
- schedulers don't take care of locality of data.

Usually

- scheduling and replication are two independent processes,
- replication has to be done by users,
- schedulers don't take care of locality of data.

Usually

- scheduling and replication are two independent processes,
- replication has to be done by users,
- schedulers don't take care of locality of data.

An idea :

Usually

- scheduling and replication are two independent processes,
- replication has to be done by users,
- schedulers don't take care of locality of data.

An idea :

Join scheduling of computation and replication of data

Agenda

- 1 Motivations
- 2 Model
 - Parameters
 - Constraints
 - Solutions
- 3 Simulations et results
- 4 Conclusions

Things we know

- platform

- ▶ n computation servers P_i
 - ★ storage space : m_i
 - ★ computation power : w_i

- bioinformatic

- ▶ m databanks d_j of size : $size_j$
- ▶ p algorithms a_k :
 - ★ linear with size of databanks : $\alpha_k * size + c_k$
- ▶ requests $R(k, j)$
 - ★ usage frequency : $f(k, j)$

Things we know

- platform

- ▶ n computation servers P_i
 - ★ storage space : m_i
 - ★ computation power : w_i

- bioinformatic

- ▶ m databanks d_j of size : $size_j$
- ▶ p algorithms a_k :
 - ★ linear with size of databanks : $\alpha_k * size + c_k$
- ▶ requests $R(k, j)$
 - ★ usage frequency : $f(k, j)$

Things we know

- platform

- ▶ n computation servers P_i
 - ★ storage space : m_i
 - ★ computation power : w_i

- bioinformatic

- ▶ m databanks d_j of size : $size_j$
- ▶ p algorithms a_k :
 - ★ linear with size of databanks : $\alpha_k * size + c_k$
- ▶ requests $R(k, j)$
 - ★ usage frequency : $f(k, j)$

Things we have to determine

- TP : throughput
- δ_i^j : placement of databanks
- $n_i(k, j)$: requests done by each server
 - ▶ number of jobs $R(k, j)$ done by P_i : $n_i(k, j)$

Constraint

- each data at least on one server
- a server cannot store more than available space
- a server cannot compute more than available computation power
- a request can be executed only if data is on the server
- job distribution follow usage frequency

Constraint

- each data at least on one server
- a server cannot store more than available space
- a server cannot compute more than available computation power
- a request can be executed only if data is on the server
- job distribution follow usage frequency

Constraint

- each data at least on one server
- a server cannot store more than available space
- a server cannot compute more than available computation power
- a request can be executed only if data is on the server
- job distribution follow usage frequency

Constraint

- each data at least on one server
- a server cannot store more than available space
- a server cannot compute more than available computation power
- a request can be executed only if data is on the server
- job distribution follow usage frequency

Constraint

- each data at least on one server
- a server cannot store more than available space
- a server cannot compute more than available computation power
- a request can be executed only if data is on the server
- job distribution follow usage frequency

Constraint

- each data at least on one server
- a server cannot store more than available space
- a server cannot compute more than available computation power
- a request can be executed only if data is on the server
- job distribution follow usage frequency

Goal : Maximize throughput of the platform (makespan)

Linear program

linear program formulation

MAXIMIZE TP ,

WITH CONSTRAINTS

$$\left\{ \begin{array}{ll} (1) \sum_{j=1}^n \delta_i^j \geq 1 & 1 \leq i \leq m \\ (2) \sum_{j=1}^n \delta_i^j \cdot size_j \leq m_i & 1 \leq i \leq m \\ (3) \sum_{k=1}^p \sum_{j=1}^n n_i(k, j) (\alpha_k * size_j + c_k) \leq w_i & 1 \leq i \leq m \\ (4) n_i(k, j) \leq v_{k,j} \cdot \delta_i^j \cdot \frac{w_i}{\alpha_k \cdot size_j + c_k} & 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq k \leq p \\ (5) \sum_{i=1}^m n_i(k, j) = f_{k,j} \cdot TP & 1 \leq i \leq m, 1 \leq j \leq n \end{array} \right.$$

A word on solution

Integer and rational number problem

- use of integer approximation for δ_i^j

With realistic information, we can notice :

- the most used data are more replicated
- storage space is not full

Agenda

- 1 Motivations
- 2 Model
- 3 Simulations et results**
 - Experimental environment
 - Results
- 4 Conclusions

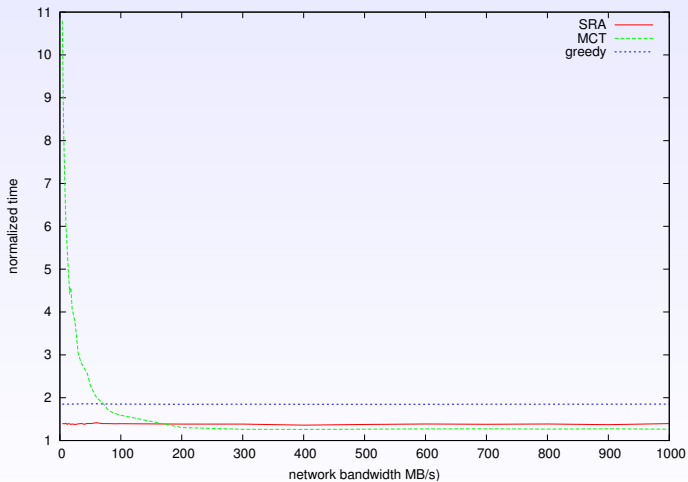
Simulation

- use of OptorSim
 - ▶ simulator for grid data management
 - ▶ developed for European DataGrid project
- largely modified to match our needs
 - ▶ heterogeneous compute system
 - ▶ batch scheduler
 - ▶ heterogeneous computation time
 - ▶ ...

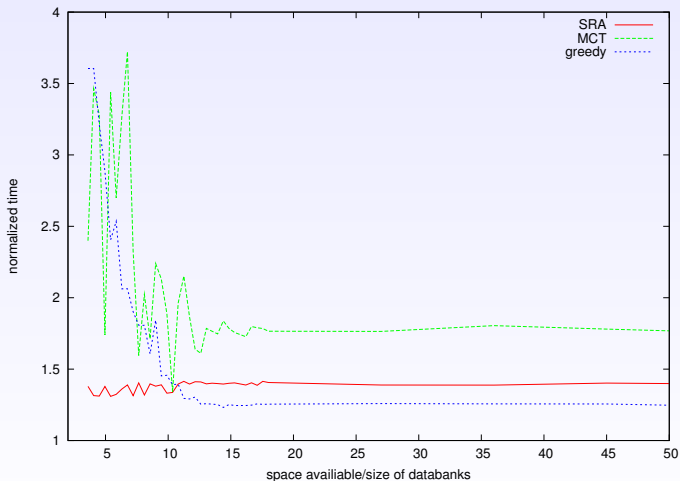
Tests

- test platform :
 - ▶ generated by Tiers
 - ▶ 10 platforms
- Requests :
 - ▶ based on real requests
 - ▶ 40000 requests

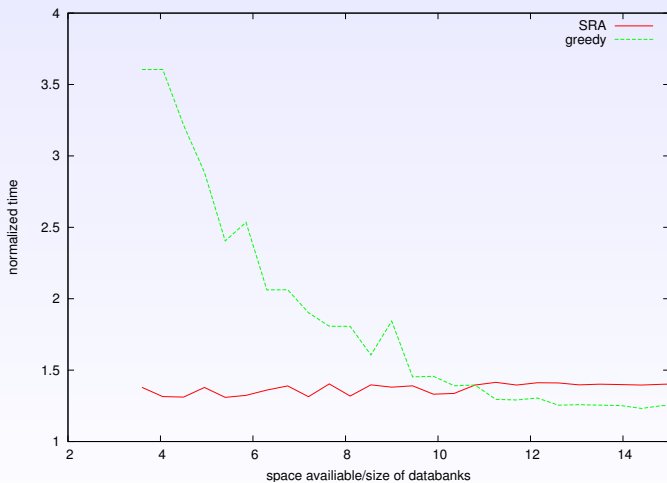
Execution time : fonction de network bandwidth



Execution time : function of storage space available



Execution time : function of storage space (zoom)



Agenda

- 1 Motivations
- 2 Model
- 3 Simulations et results
- 4 **Conclusions**
 - What have been done...
 - What is to be done...

Conclusions

- Steady state model
- Simulation
- Good optimisations for
 - ▶ low speed network
 - ▶ small storage space
- Placement is efficient

Work in progress

- Dynamic solution
- Real execution with DIET environment