Data Location-Aware Job Scheduling in the Grid
Application to the GridWay Metascheduler

Antonio Delgado Peris (CIEMAT, Spain), Jose Hernández (CIEMAT, Spain), Eduardo Huedo (Universidad Complutense de Madrid, Spain), Ignacio M. Llorente (Universidad Complutense de Madrid, Spain)

Data Location-Aware Job Scheduling

Agreed in grid community, data location needs to be considered for job scheduling:

To avoid lost time waiting for input data staging

Approaches to this problem:

1. Sending jobs to the sites holding the input data
   - A. May be suboptimal when these sites are busy or inaccessible
   - B. Requires independent data management
2. Balancing data transfer time and expected job delay time to select job destination
   - A. Requires estimation of the transfer costs: e.g. *Network Weather Service*
   - B. In general, does not consider restrictions like VO policies, limited storage space
   - C. More complex and costly when calculations are made for a whole grid
3. For 1 and 2, sometimes automatic replication of files is suggested

Additional problems with data replication:

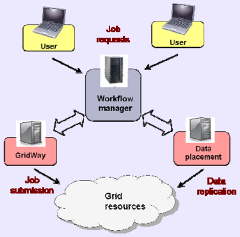
- Storage elements get filled
- VO policies on storage need to be considered when scheduling
- Competition for network and storage resources is increased

Some conclusions:

- Jobs to data much better than ignoring location
- Optimum scheduling only possible considering data location and computing resources characteristics
  - This is very difficult to achieve in practice
- Necessities, constraints vary from VO to VO
  - Need a flexible system (configurable policy)
- Other reasons to minimize data transfers exist
- It is better to decouple job scheduling and data transfers: let placement systems manage these

Towards an optimum VO-global scheduling strategy:

- Optimal scheduling of single job does not guarantee optimal global scheduling
- Coordinated management of data placement and job scheduling is required
- A workflow manager with a global view of a VO might know better
- It would have to maintain a queue of jobs and schedule data movements and job submissions according to global necessities



The GridWay Prototype

GridWay:

- General purpose metascheduler
- By the Distributed Systems Architecture Group of the University Complutense of Madrid
- Full Globus project

Current use of data location information:

- Not consider when scheduling
- Only best computer resource

New prototype:

- Possibility to take into account presence of data both in requirements and rank
- This is more flexible than glite WMS
- VO/users can set the policy to use
- Does not include transfers latency estimation

Implementation:

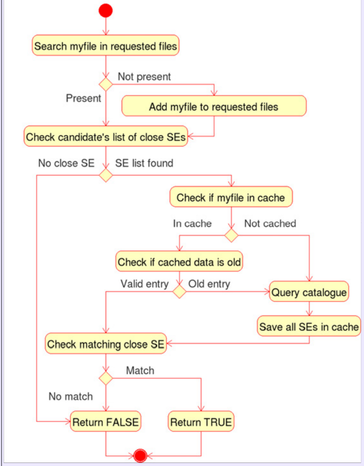
- EGEE information plugin modified: retrieve closeSE attributes
- Interface modified: data functions in requirement and rank expressions
- Daemon modified: query to specified catalogue using DLI interface
- Cache of locations to minimize number of catalogue interactions
- Remote files list available to user in job's environment

`DATA_CATALOG`	Job template var
`CLOSE_DATA(file)`	Requirem. expression

`HAS_CLOSE_DATA(file)`
`SIZE_CLOSE_DATA(file)`
	Rank expressions

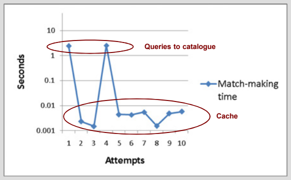
`GW_CLOSE_SE` `GW_REMOTE_FILES`	Job environment vars

Rank expression: HAS_CLOSE_DATA(myfile)

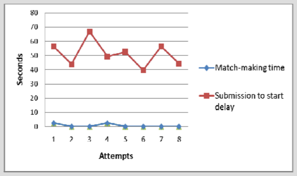


Catalogue query delay

Match-making time

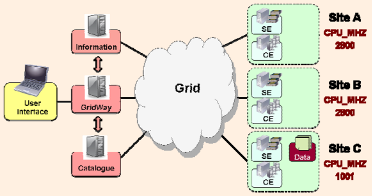


Submission delay vs match-making



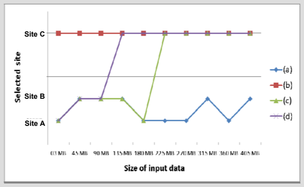
Testing

Policy testing

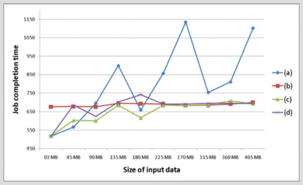


Policy		Requirem.	Rank
(a)	Don't use data loc.	–	CPU_MHZ
(b)	Jobs to data	CLOSE_DATA ("file")	CPU_MHZ
(c)	Balance transfer time and CPU	–	CPU_MHZ + Kt * SIZE_CLOSE_DATA("file")
(d)	(c) + extra transfer penalty	–	CPU_MHZ + Kp * Kt * SIZE_CLOSE_DATA("file")
Kt ≡ Transfer-CPU balance factor = 0.0075 *Kp ≡ Transfer penalty factor = 2*			

Destination by algorithm and input data



Job turnaround time by algorithm



Algorithm	Average Time	Transfers
(a)	799.4	10
(b)	687.4	0
(c)	648.4	5
(d)	674.0	3

(a) obtains the worst results

(b) avoids all data transfers

(c) achieves the best average time

(d) accepts slightly worst average time to reduce transfers