Scheduling Tasks Sharing Files from Distributed Repositories

Arnaud Giersch[1], Yves Robert[2] and Frédéric Vivien[2]

[1] ICPS/LSIIT, University Louis Pasteur, Strasbourg, France
[2] École normale supérieure de Lyon, France

September 1, 2004

Talk overview

Talk overview

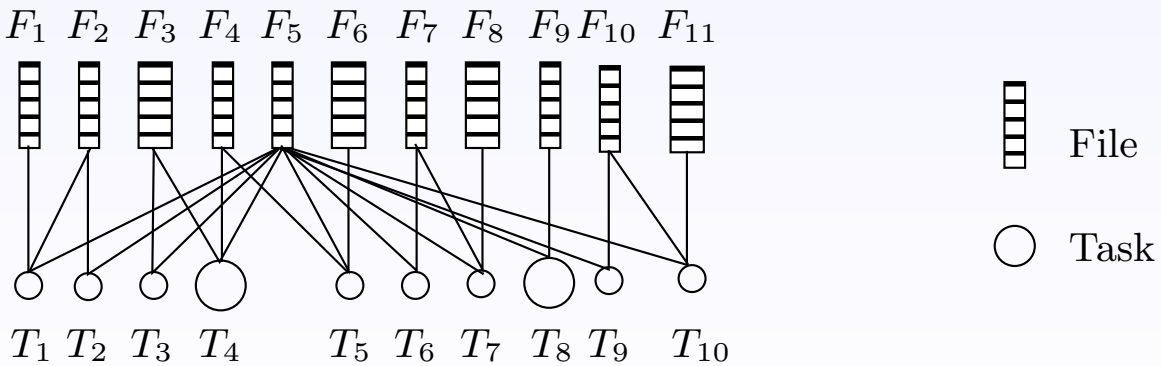
The tasks and the files

Tasks

- Independent from each other
- Have different durations
- Depend from input files

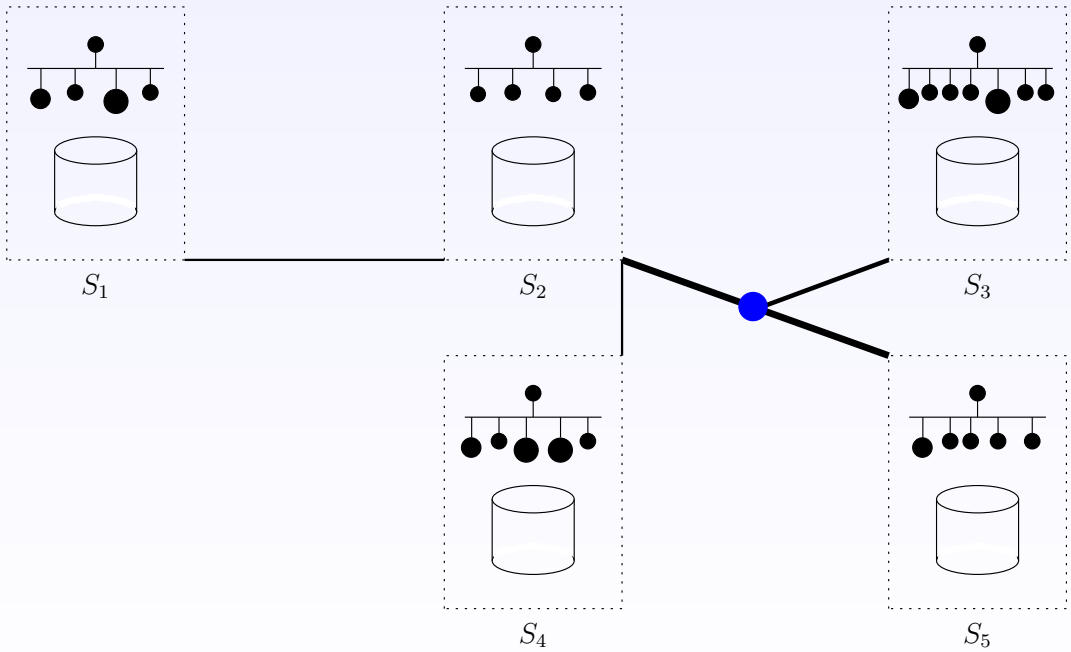
Files

- Have different sizes
- Several tasks may depend on a same file



The platform

- A platform = A network of servers
- A server = A local repository + A computational cluster
- The communication network is heterogeneous
- The clusters are not identical



The problem

Question: How to map and schedule the tasks?

Objective: minimize the overall execution time.

Constraint: For the server $\mathcal{S}$ to execute the task T: all the files T depends upon must be stored in the repository of $\mathcal{S}$ *before* $\mathcal{S}$ can start processing T.

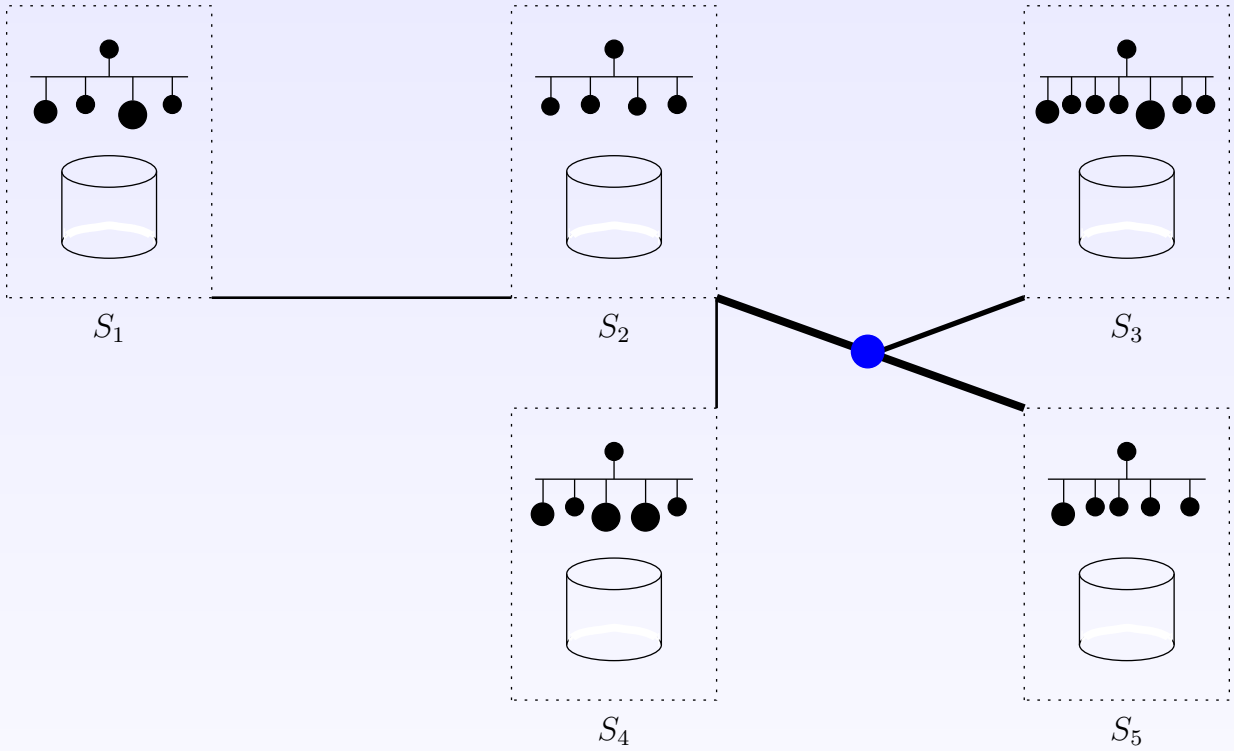
The problem

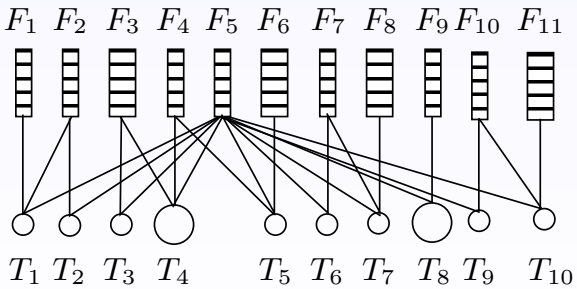
Question: How to map and schedule the tasks?

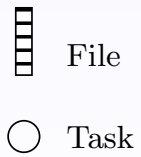
Objective: minimize the overall execution time.

Constraint: For the server $\mathcal{S}$ to execute the task T: all the files T depends upon must be stored in the repository of $\mathcal{S}$ *before* $\mathcal{S}$ can start processing T.

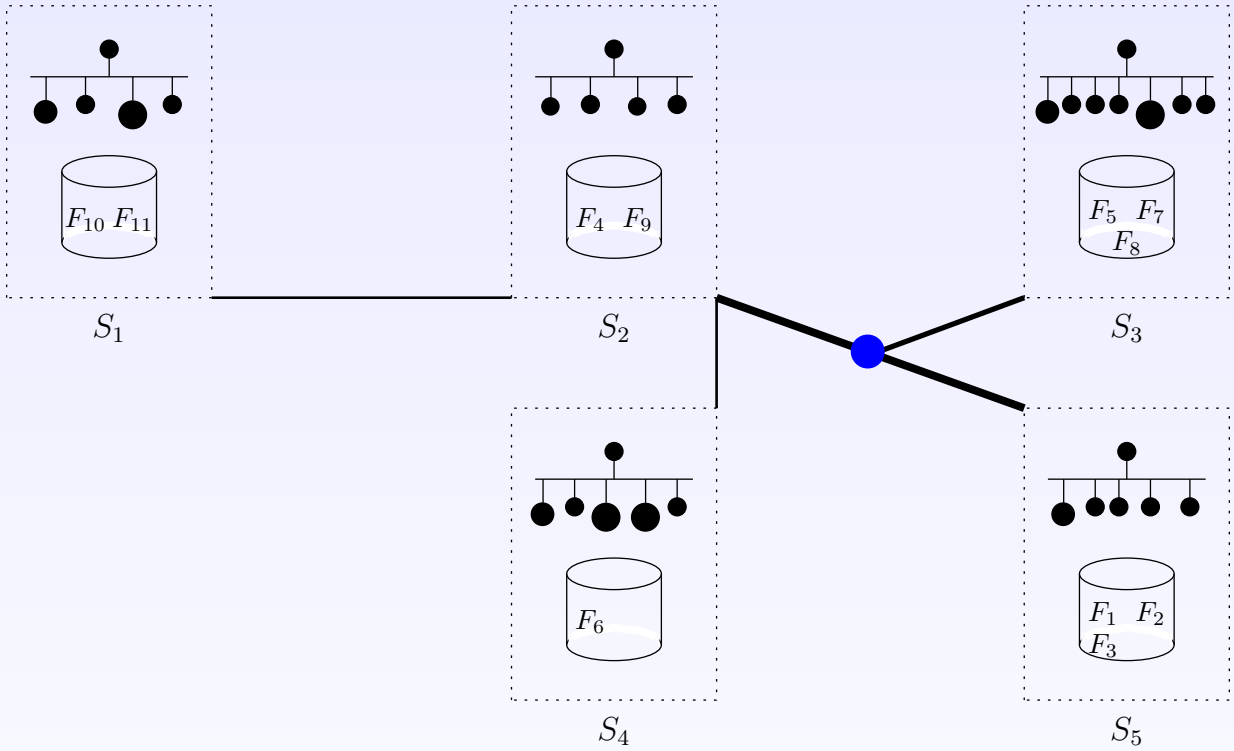
Processing an example

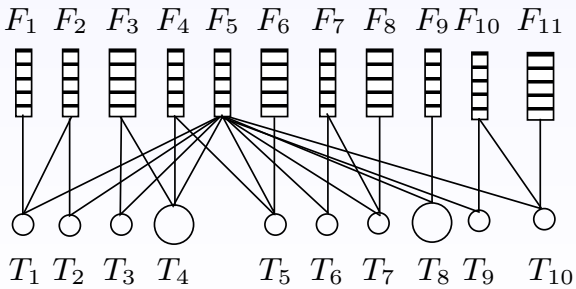


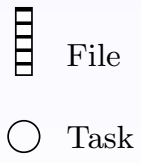




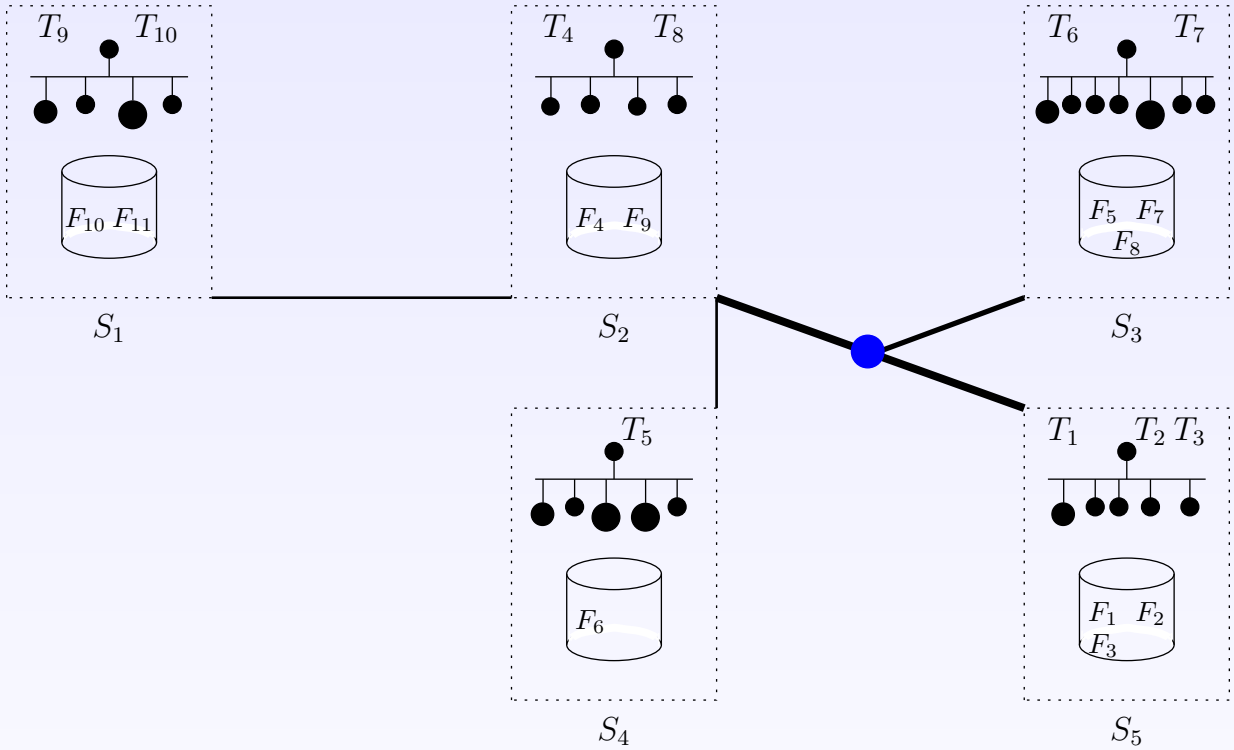
Processing an example

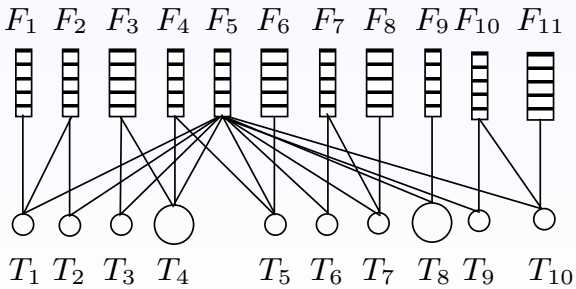


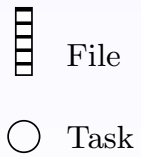




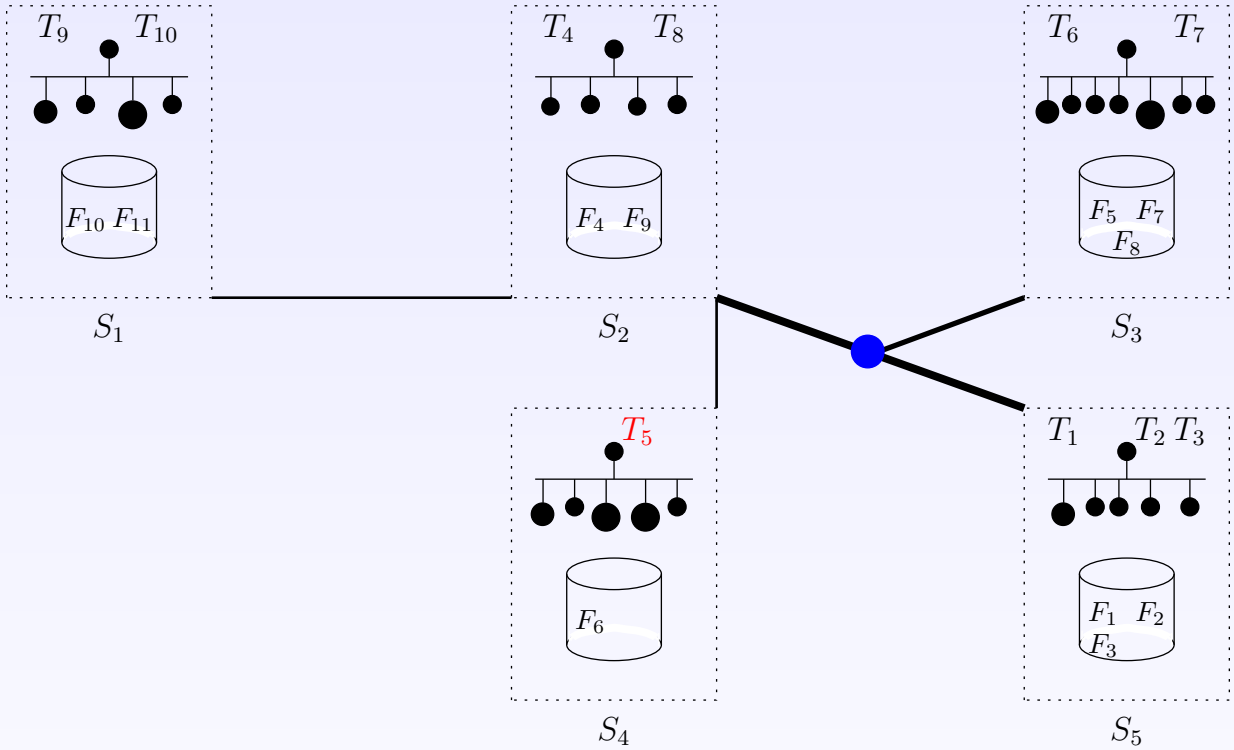
Processing an example

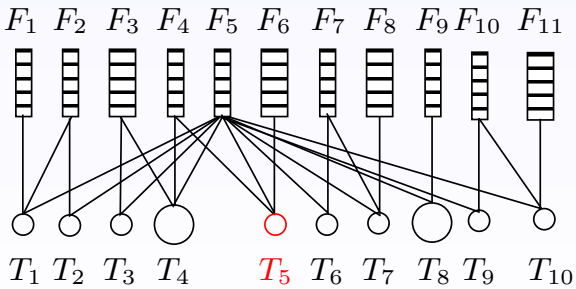


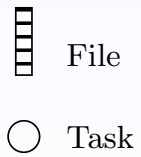




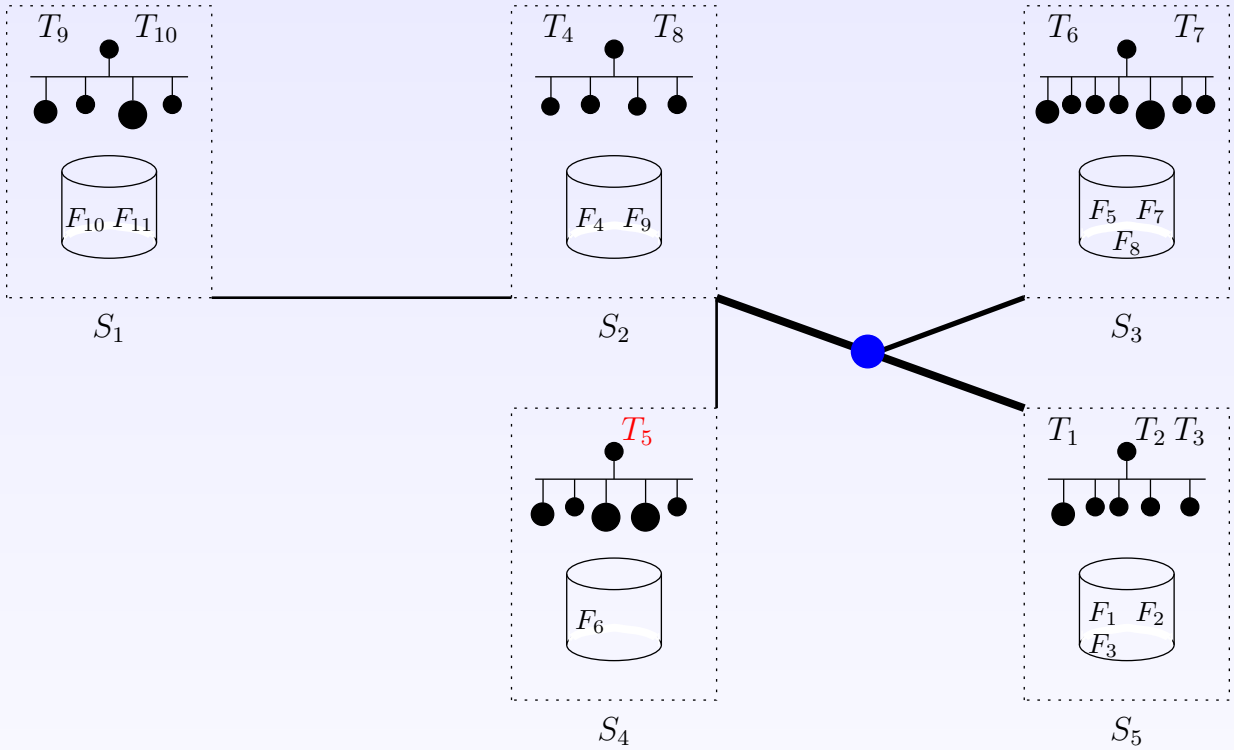
Processing an example

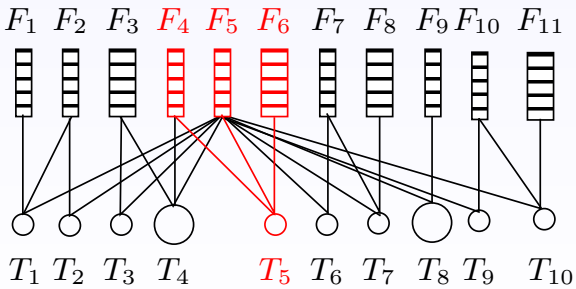


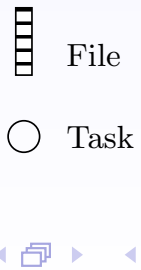




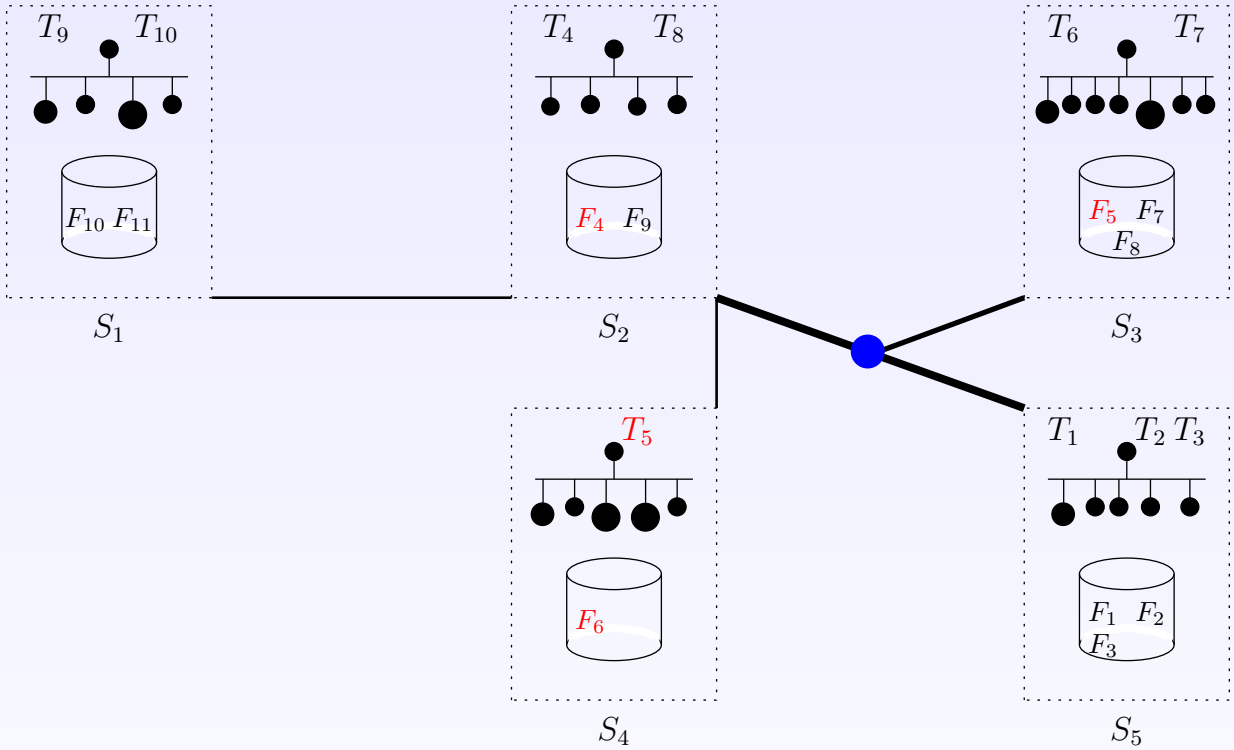
Processing an example

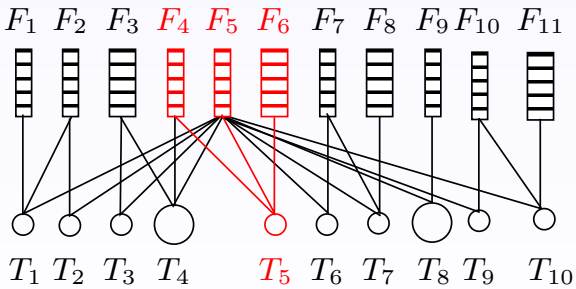




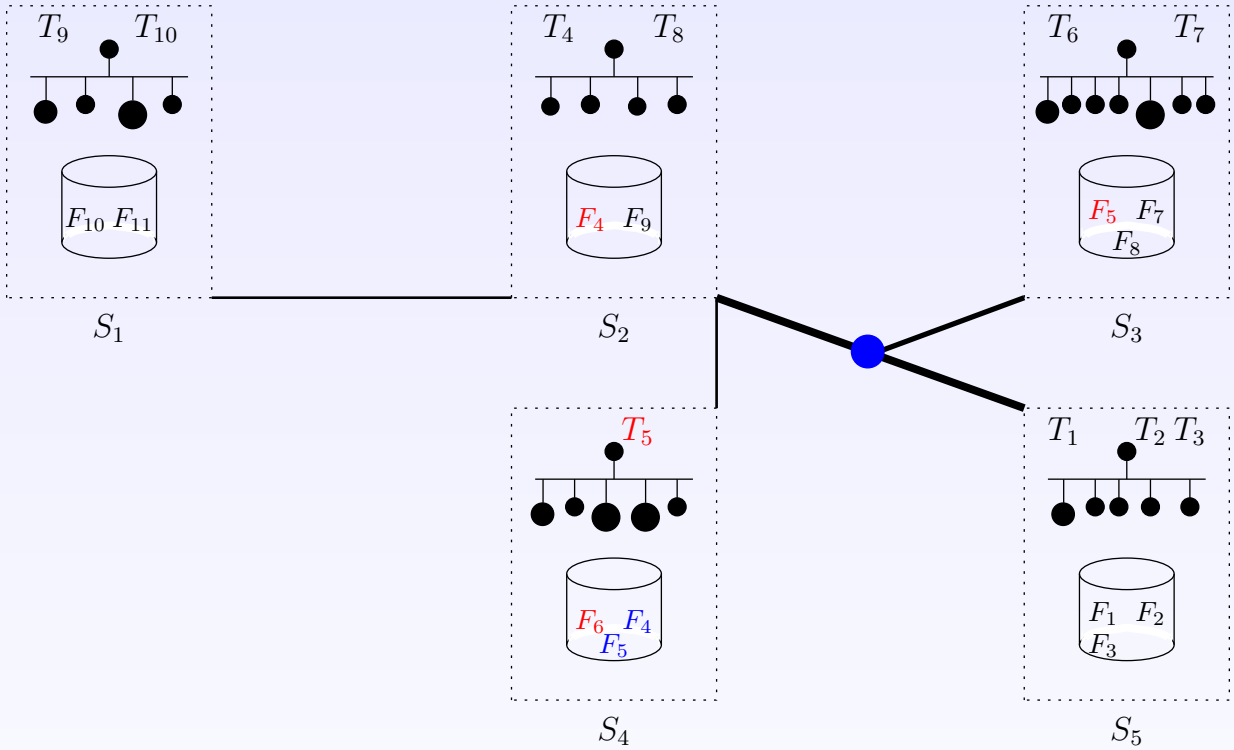


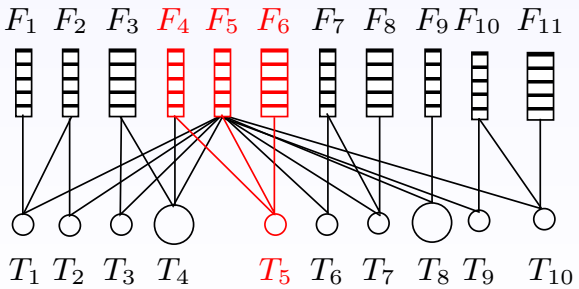
Processing an example

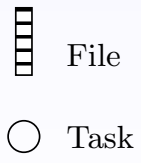




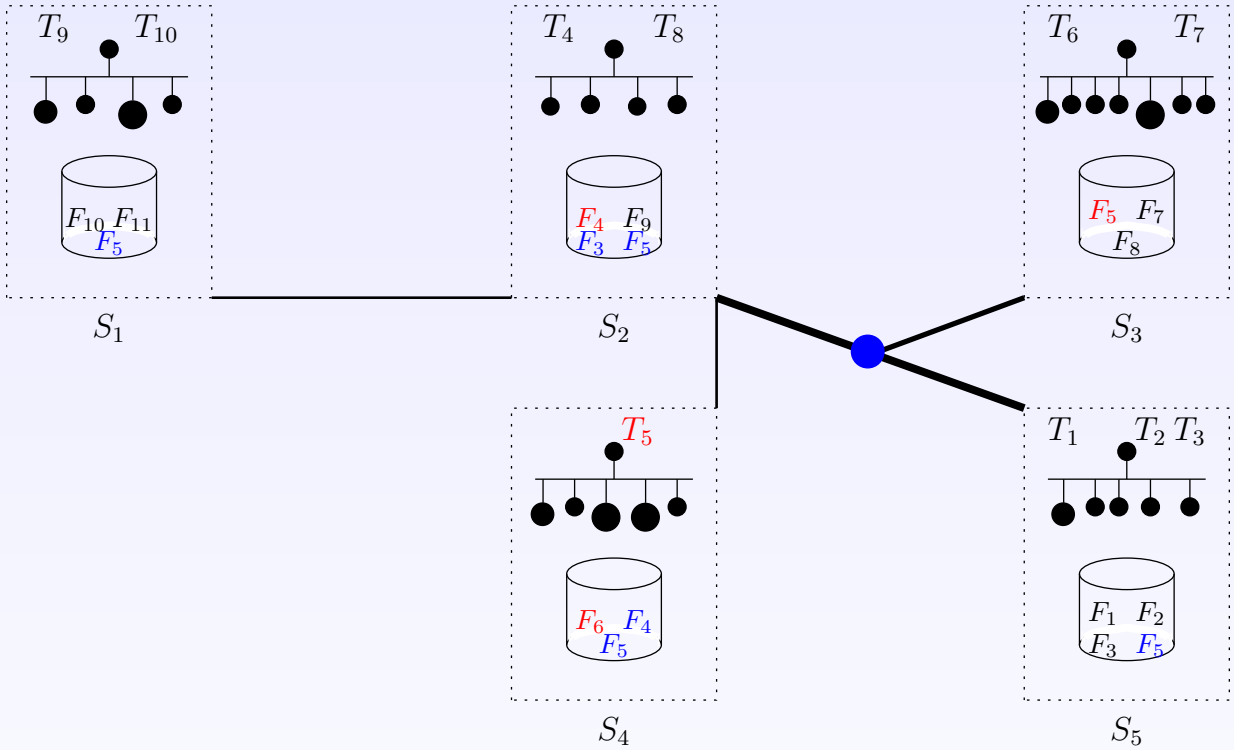
Processing an example

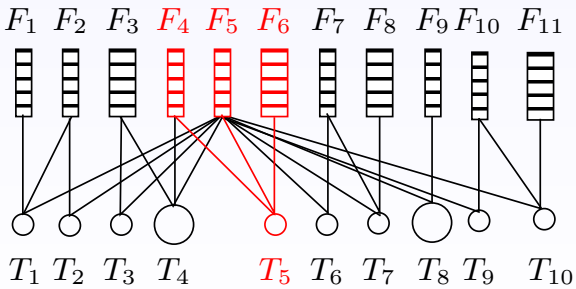


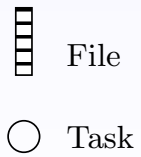




Processing an example







Complexity

Input A bipartite application graph
A platform graph

Hypotheses Uniform file sizes
Homogeneous interconnection network
Zero processing time

Question Given a bound K is it possible to schedule all tasks with K time units?

Complexity NP-complete

Complexity

Input A bipartite application graph
A platform graph

Hypotheses Uniform file sizes
Homogeneous interconnection network
Zero processing time

Question Given a bound K is it possible to schedule all tasks with K time units?

Complexity NP-complete

Complexity

Input A bipartite application graph
A platform graph

Hypotheses Uniform file sizes
Homogeneous interconnection network
Zero processing time

Question Given a bound K is it possible to schedule all tasks with K time units?

Complexity NP-complete

Complexity

Input A bipartite application graph
A platform graph

Hypotheses Uniform file sizes
Homogeneous interconnection network
Zero processing time

Question Given a bound K is it possible to schedule all tasks with K time units?

Complexity NP-complete

Talk overview

The min-min heuristic

Motivation: validated on master-slave systems

While there are unscheduled tasks

1. For each unscheduled task T_k
   - For each processor $C_{i,j}$
     Evaluate the MCT of T_k if mapped on $C_{i,j}$
2. Pick couple $(C_{i,j}, T_k)$ with minimum MCT
3. Schedule T_k on $C_{i,j}$

Problem: How to evaluate the MCT?

Difficulty: Scheduling the communications is NP-complete!

The min-min heuristic

Motivation: validated on master-slave systems

While there are unscheduled tasks

1. For each unscheduled task T_k
   - For each processor $C_{i,j}$

     Evaluate the MCT of T_k if mapped on $C_{i,j}$
2. Pick couple $(C_{i,j}, T_k)$ with minimum MCT
3. Schedule T_k on $C_{i,j}$

Problem: How to evaluate the MCT?

Difficulty: Scheduling the communications is NP-complete!

The min-min heuristic

Motivation: validated on master-slave systems

While there are unscheduled tasks

1. For each unscheduled task T_k
   - For each processor $C_{i,j}$

     Evaluate the MCT of T_k if mapped on $C_{i,j}$
2. Pick couple $(C_{i,j}, T_k)$ with minimum MCT
3. Schedule T_k on $C_{i,j}$

Problem: How to evaluate the MCT?

Difficulty: Scheduling the communications is NP-complete!

The min-min heuristic

Motivation: validated on master-slave systems

While there are unscheduled tasks

1. For each unscheduled task T_k
   - For each processor $C_{i,j}$

     Evaluate the MCT of T_k if mapped on $C_{i,j}$
2. Pick couple $(C_{i,j}, T_k)$ with minimum MCT
3. Schedule T_k on $C_{i,j}$

Problem: How to evaluate the MCT?

Difficulty: Scheduling the communications is NP-complete!

Scheduling the communications

Hypothesis: fixed routing.

Greedy scheduling: a new communication is scheduled on a network link *after* the already scheduled communications which use that link.

Insertion scheduling: for each link, we memorize the date and length of each communication and we use a first-fit algorithm
More precise but more expensive...

Scheduling the communications

Hypothesis: fixed routing.

Greedy scheduling: a new communication is scheduled on a network link *after* the already scheduled communications which use that link.

Insertion scheduling: for each link, we memorize the date and length of each communication and we use a first-fit algorithm
More precise but more expensive...

Scheduling the communications

Hypothesis: fixed routing.

Greedy scheduling: a new communication is scheduled on a network link *after* the already scheduled communications which use that link.

Insertion scheduling: for each link, we memorize the date and length of each communication and we use a first-fit algorithm
More precise but more expensive...

Scheduling the communications

Hypothesis: fixed routing.

Greedy scheduling: a new communication is scheduled on a network link *after* the already scheduled communications which use that link.

Insertion scheduling: for each link, we memorize the date and length of each communication and we use a first-fit algorithm
More precise but more expensive...

Complexity of the adapted min-min scheme

$$O(n_{\text{tasks}}^2 \cdot n_{\text{servers}} \cdot (\Delta T \cdot n_{\text{servers}} + O_c) + n_{\text{tasks}} \cdot \max_{1 \leq i \leq n_{\text{servers}}} c_i).$$

(with greedy communication scheduling)

Complexity of the adapted min-min scheme

$$O(n_{\text{tasks}}^2 \cdot n_{\text{servers}} \cdot (\Delta T \cdot n_{\text{servers}} + O_c) + n_{\text{tasks}} \cdot \max_{1 \leq i \leq n_{\text{servers}}} c_i).$$

(with greedy communication scheduling)

Talk overview

Principle of the new heuristics

While there are unscheduled tasks

1. For each cluster C_i pick the "best" unscheduled candidate task T_k
2. Pick the "best" couple (C_i, T_k)
3. Schedule T_k on C_i

"best" defined by a *cost* function: estimation of MCT

cost function: MCT not considering scheduled computations and communications

Static heuristics

Principle of the heuristics

1. Local candidate for cluster C_i:
   unscheduled task T_k of lowest *cost*
2. We pick the couple (C_i, T_k) of lowest *cost*
3. We schedule the communications needed to run T_k on C_i
4. We schedule T_k on C_i

Complexity: the n_{tasks}^2 is replaced by a $n_{\text{tasks}} \log(n_{\text{tasks}})$ term

MCT variant

Two improvements:

1. If all files the unscheduled task T_k depends upon are on R_i, T_k is the local candidate of C_i (whatever its *cost*)
2. We compute the MCT of the local candidate of each cluster, and pick the task of minimum MCT

Dynamic heuristics

Motivation: the *cost* estimate becomes less and less pertinent during the execution

Workaround: dynamic *costs*: each time a file F is duplicated we only update the *costs* of the tasks depending on F

`dynamic1`: for each cluster, a heap of dynamic *costs*

`dynamic2`: we select k tasks of lowest dynamic *costs* per cluster

Talk overview

Simulations: the Platforms

- 7 servers
- Server network: clique, random tree, or ring
- Each cluster made of 8, 16, or 32 processors
- Processor speeds randomly picked from a set of actual values
- Communication link bandwidths randomly picked from a set of actual values

Communication to computation cost ratio

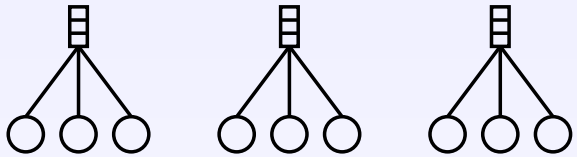
Normalized processor and communication average characteristics in order to model three main types of problems:

- computation intensive (ratio=0.1);
- communication intensive (ratio=10); and
- intermediate (ratio=1).

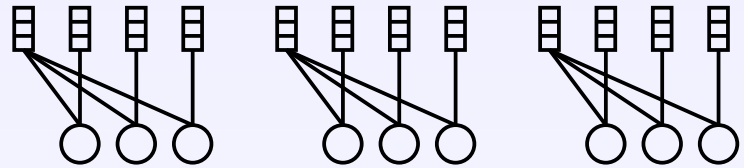
Simulations: the File-Task Graphs

Files and tasks sizes are randomly and uniformly taken in $[0.5; 5]$.

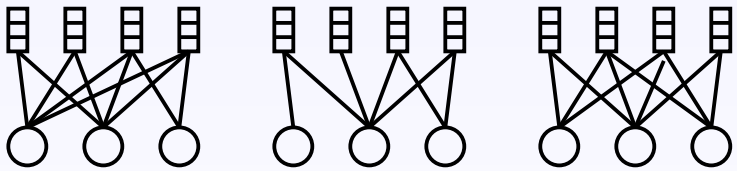
Initial distribution of files over repository: random



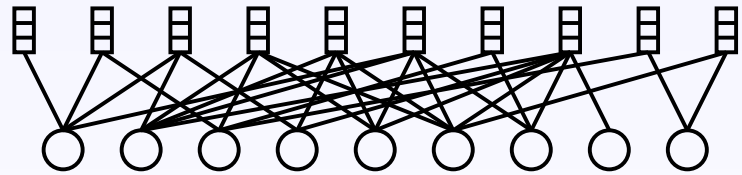
Forks (100 $\times$ 20 tasks/file)



Two-one (2 files/task)



Partitioned (20 cc, 1–10 files/task)



Random (1–50 files/task)

Each graph: 1500 tasks and 1750 files (except for forks: 70 files).

Simulations: Evaluations

- Two characteristics per heuristic:
  performance: simulated end time of the schedule;
  cost: time used to construct the schedule.
- Absolute values obtained from different configurations cannot be compared

$$\text{relative performance}(H, \text{conf}) = \frac{\text{performance}(H, \text{conf})}{\min_{h \in \{\text{heuristics}\}} \text{performance}(h, \text{conf})}$$

- The same applies for *relative costs*.

Results: relative performances and costs

Heuristic	Basic version		mct variant		mct+insert variant	
	Perf.	Cost	Perf.	Cost	Perf.	Cost
min-min	1.14	31,050	-	-	1.08	61,711
sufferage	1.16	33,985	-	-	1.07	77,991
static	2.20	16	1.46	44	1.18	56
dynamic 1	2.32	42	1.35	67	1.11	77
dynamic 2	2.42	310	1.92	82	1.40	90
randommap	141	11	1.32	41	1.08	53

Each value is an average over 36,000 simulations.

Conclusion

- Adapted the `min-min` heuristic to our framework
- Designed new heuristics with reasonably good performances and low computational costs

  Best heuristic: schedules whose makespan is 1.1% longer than those of the best `min-min` variant; produces them 585 times faster than the quickest variant of `min-min`