Handling Collusion in Dynamic Desktop Grids

Louis-Claude CANON, Emmanuel JEANNOT and Jon WEISSMAN

Project-Team AlGorille
Loria/INRIA/Université Henri Poincaré
University of Minesota

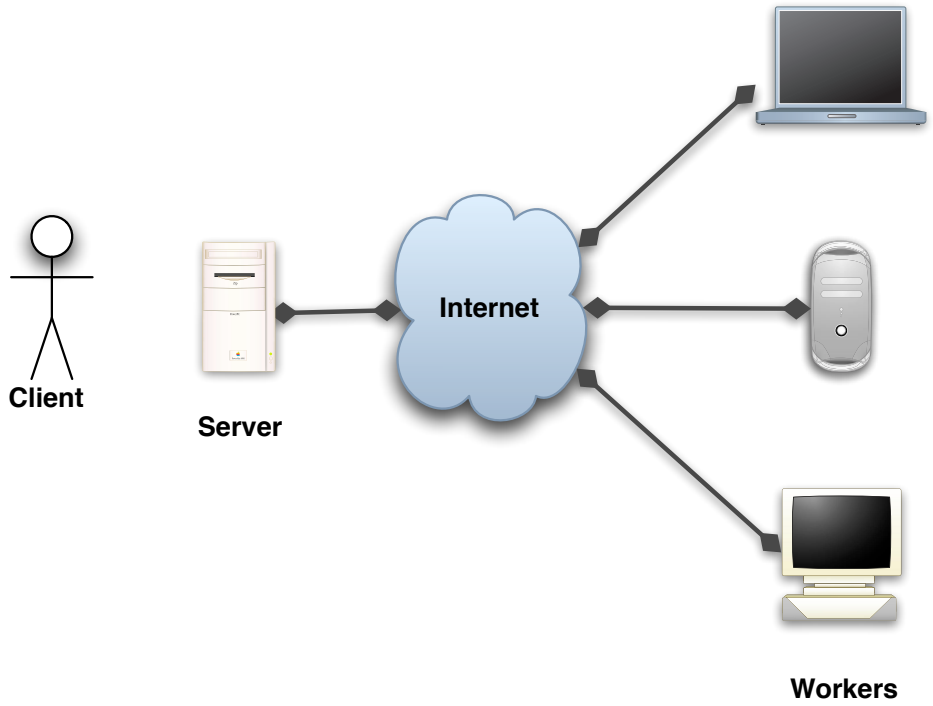
ALEAE Project

August 25, 2009

Outline

Outline

Principles of Desktop Grid



Distributed Computing

The server assigns **job** ($j \in J$) to each active **worker** ($w \in W$).
Each worker returns a **result** ($r \in R$) for the corresponding job.
The client wants to have the *correct* result for each submitted job.
Example: SETI@Home, BOINC-based projects.

Cheating

Byzantine fault

Unreliability or malicious isolated peer (worker).
35% of workers gives at least one wrong result (Derrick Kondo *et al.* 2007).
Allow to increase credit for example.

Cheating

Byzantine fault

Unreliability or malicious isolated peer (worker).
35% of workers gives at least one wrong result (Derrick Kondo *et al.* 2007).
Allow to increase credit for example.

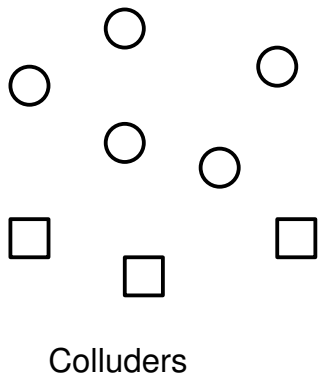
k-majority

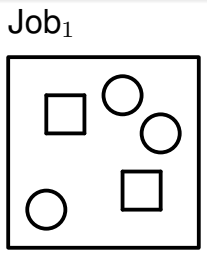
Assign a job to *k* distinct peers (redundancy) and select the result that has the majority.
BOINC use a quorum-based system.

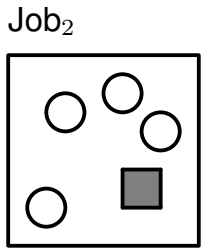
Organized cheaters

Collusion

A subset of the workers produces the same wrong result for some given job.
Against Sybil attack or library with platform specific bugs.







Outline

Mechanisms

Generality

Generic and non-intrusive mechanism: no information available on the structure of the jobs and their results.
No trustworthy computation resources.
Blacklisting is not necessary (risk of churn and whitewashers).

Mechanisms

Generality

Generic and non-intrusive mechanism: no information available on the structure of the jobs and their results.
No trustworthy computation resources.
Blacklisting is not necessary (risk of churn and whitewashers).

Main components

Resource grouping assures duplications of jobs by creating a set of workers for each job.

Result certification selects one of the results as the best one.

Mechanisms

Generality

Generic and non-intrusive mechanism: no information available on the structure of the jobs and their results.
No trustworthy computation resources.
Blacklisting is not necessary (risk of churn and whitewashers).

Main components

Resource grouping assures duplications of jobs by creating a set of workers for each job.

Result certification selects one of the results as the best one.

Objectives (Zhao and Lo, 2005)

Overhead minimum redundancy.

Accuracy maximum of correct certified results.

Grouping strategies

Detection orientation

Large groups with workers with unknown interactions (exploration).

Grouping strategies

Detection orientation

Large groups with workers with unknown interactions (exploration).

Avoidance orientation

Group with workers belonging to distinct colluding sets (exploitation).

Grouping strategies

Detection orientation

Large groups with workers with unknown interactions (exploration).

Avoidance orientation

Group with workers belonging to distinct colluding sets (exploitation).

Need for informations

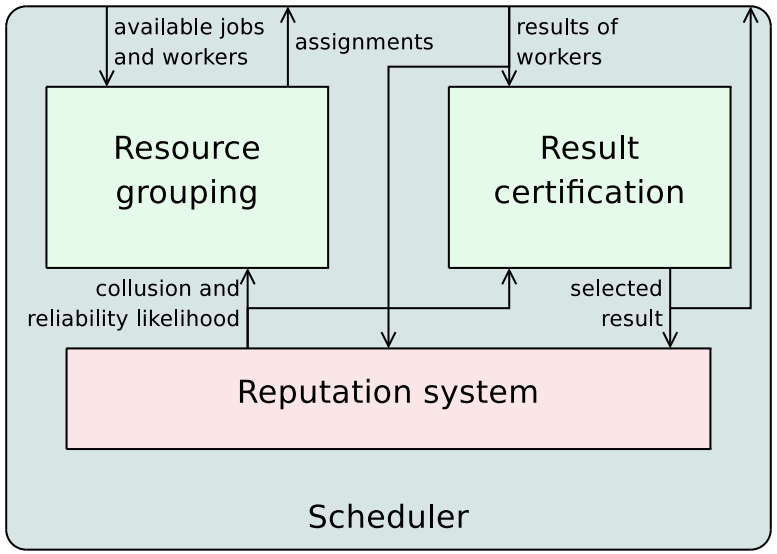
In either cases, the probability to collude of a set of workers is needed.
More generally, the colluding characteristics of the system.

Outline

Reputation system: an underlying problem

Problem description

Based only on the generated results, what can we say?
Build a **reputation system** that will then be used at higher level.



Objectives of the reputation system

Goal

Determine the probability of collusion of any subset of workers.

Objectives of the reputation system

Goal

Determine the probability of collusion of any subset of workers.

Criteria (Jøsang, Ismail, and Boyd, 2007)

Accuracy Root Mean Square Deviation (RMSD) of all generated informations

Adaptativeness or dynamicity #iterations to acknowledge changes

Robust resistant to attacks

Smooth aggregated difference between 2 iterations

Introspection RMSD of error estimations

Observations

Input

$< t, w, j, r >$ (timestamp, worker, job, result)
$< t, j >$ whenever a job is finished

Observations

Input

$< t, w, j, r >$ (timestamp, worker, job, result)
$< t, j >$ whenever a job is finished

Observations = pairwise interactions

Considering that we observe interactions between individual workers lead to quadratic complexity for most involved algorithms and low precision for workers with low activity.

Observations

Input

$< t, w, j, r >$ (timestamp, worker, job, result)
$< t, j >$ whenever a job is finished

Observations = pairwise interactions

Considering that we observe interactions between individual workers lead to quadratic complexity for most involved algorithms and low precision for workers with low activity.

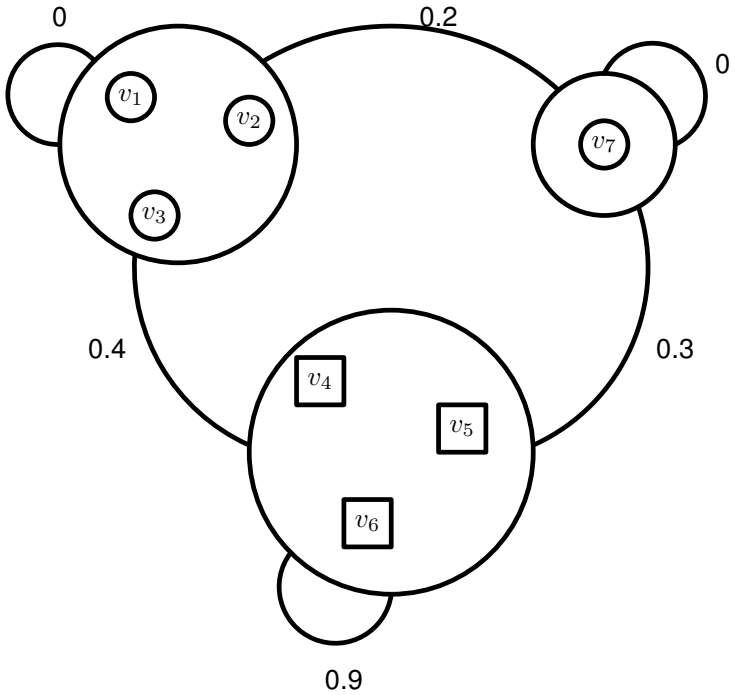
Observations = group interactions

Consider that an observation informs us on the interaction between group(s) of workers. This involves to characterize the group to which each worker belongs.

Architecture

Data structure

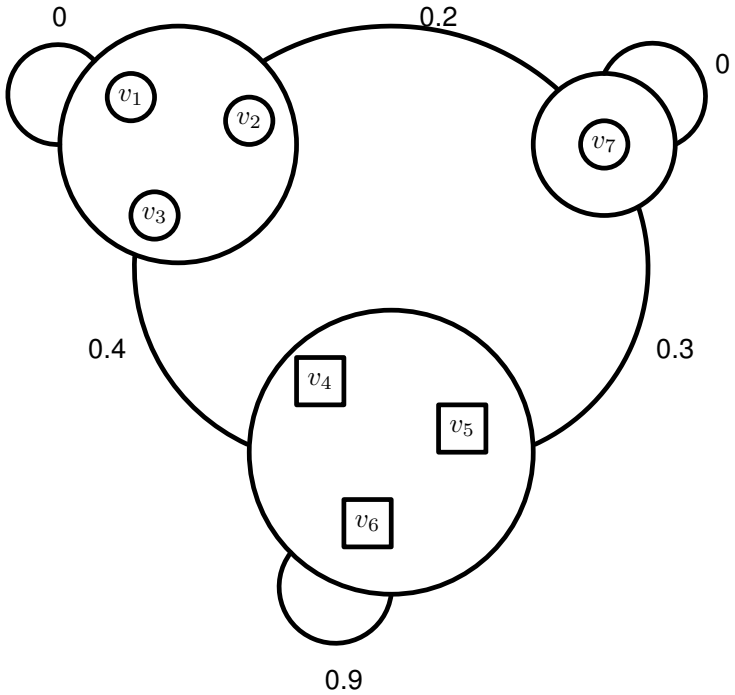
Graph: each node corresponds to a set of worker; each edge, to some collusion characteristics.



Architecture

Data structure

Graph: each node corresponds to a set of worker; each edge, to some collusion characteristics.



Algorithm

Initially, each worker is in a singleton.
We proceed by succession of merges and splits operations.

Reputation representation

Representation

- collusion likelihood between workers from set i and set j: c_{ij}
- agreement likelihood: a_{ij}

Reputation representation

Representation

- collusion likelihood between workers from set i and set j: c_{ij}
- agreement likelihood: a_{ij}

Relation

$a_{ij} = 1 + 2 \times c_{ij} - c_{ii} - c_{jj}$

$c_{ij} \leq \frac{1+a_{ij}-a_{1i}-a_{1j}}{2}$ where the index of the biggest set is 1

Reputation representation

Representation

- collusion likelihood between workers from set i and set j: c_{ij}
- agreement likelihood: a_{ij}

Relation

$a_{ij} = 1 + 2 \times c_{ij} - c_{ii} - c_{jj}$

$c_{ij} \leq \frac{1+a_{ij}-a_{1i}-a_{1j}}{2}$ where the index of the biggest set is 1

Comparison

$\{c_{ij}\} \rightarrow \{a_{ij}\}$ but $\{c_{ij}\} \nleftarrow \{a_{ij}\}$, then collusion likelihood has more informations.

Agreement likelihood is easier to manage, faster and more stable.

Example of a merging criterion

Merging sets i and j with collusion representation.
Input: c_{ii}, c_{ij} and c_{jj}

$$c_{ii} \approx c_{ij} \approx c_{jj} \wedge \min(N(c_{ii}), N(c_{ij}), N(c_{jj})) > \max(\Sigma, \frac{n}{m}) \times \frac{\alpha}{(1-d_1)(1-d_2)(1-d_3)}$$

$N(c)$ number of observations with which c is estimated

Σ sum of size of sets i and j

n number of workers

m number of characterized groups

d_k difference between c_{ii} and c_{ij}

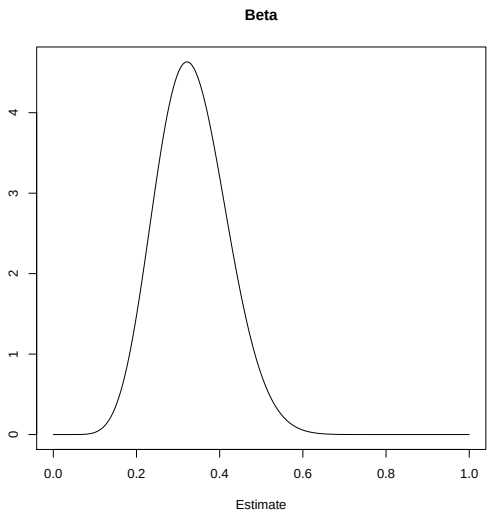
$$\alpha = \begin{cases} 2.5 & \Sigma > B \\ 1 & \text{otherwise} \end{cases}$$

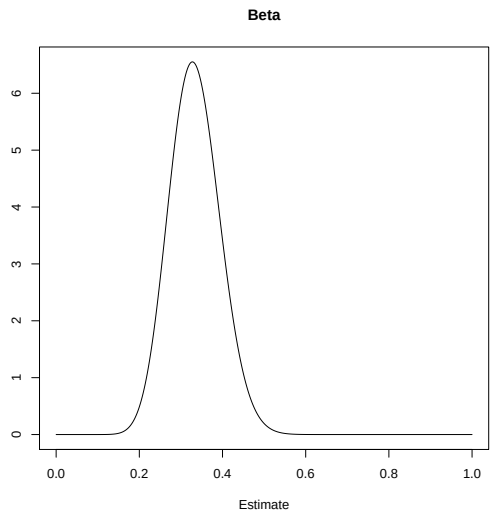
B is the size of the biggest characterized set

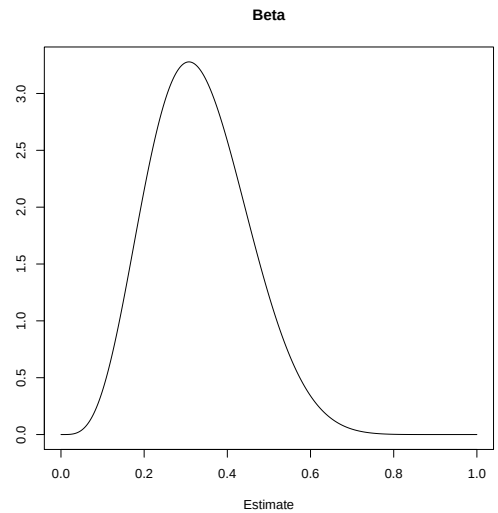
Estimator

How to deal with uncertainty (sparse observations)?

Values are modeled by random variables, *e.g.* Beta function (Jøsang, Ismail, 2002).
Arithmetic operation propagates errors (variance).







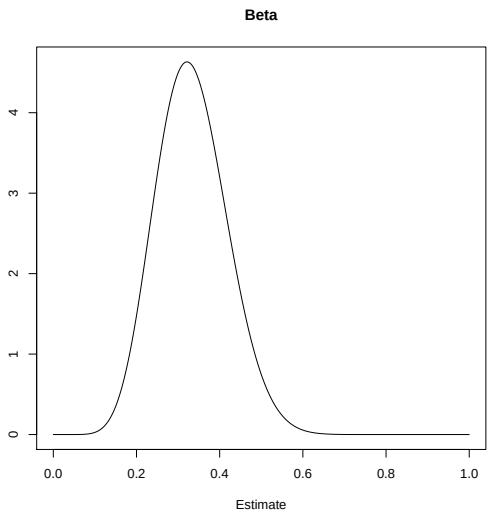
Estimator

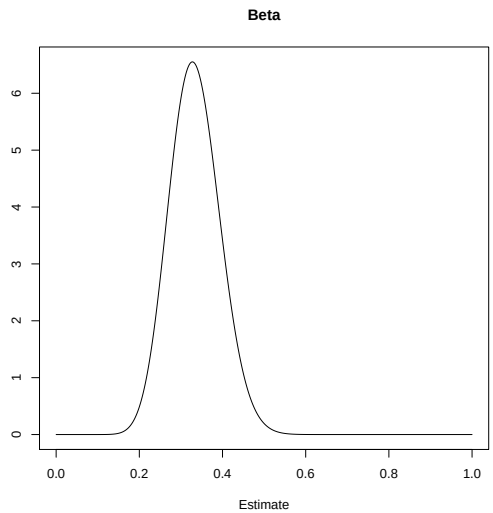
How to deal with uncertainty (sparse observations)?

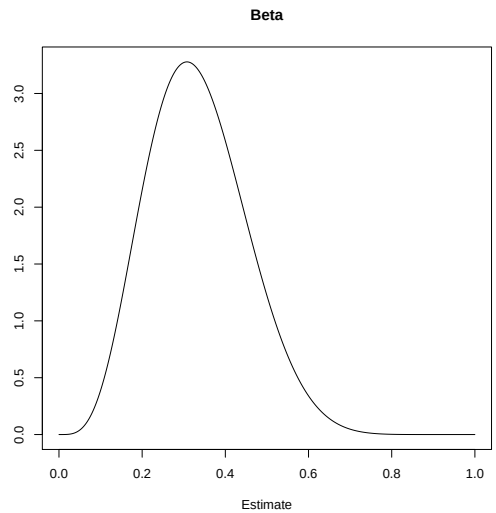
Values are modeled by random variables, *e.g.* Beta function (Jøsang, Ismail, 2002).
Arithmetic operation propagates errors (variance).

How to deal with dynamicity?

Bayesian inference is used to reinitialize each estimator when estimation is too far from recent observations.







Outline

Trace preparation (1)

No ready traces

We want $< t, w, j, r >$.
We have availibility traces (FTA, . . .), workload archive (GWA, . . .) and one performance file (FTA/SETI@home).

Trace preparation (1)

No ready traces

We want $< t, w, j, r >$.
We have availibility traces (FTA, . . .), workload archive (GWA, . . .) and one performance file (FTA/SETI@home).

Mixing distinct traces

We mix FTA and GWA trace together.
Number of needed cores for each job in GWA is eluded.
GWA is for HPC platform not Desktop grid (still better than arbitrary model).

Trace preparation (1)

No ready traces

We want $< t, w, j, r >$.
We have availibility traces (FTA, ...), workload archive (GWA, ...) and one performance file (FTA/SETI@home).

Mixing distinct traces

We mix FTA and GWA trace together.
Number of needed cores for each job in GWA is eluded.
GWA is for HPC platform not Desktop grid (still better than arbitrary model).

Other adjustment

The same performance file is used for distinct availability trace because availability and power are uncorrelated on SETI@home.

Trace preparation (2)

Scheduling

Redundancy-based scheduler: achieve a quorum of q with initial and maximal duplication l and l_{max}.
Jobs scheduled on workers when available (durations determined with the performance file and the normal duration of the job).
Result computations are based on reliability and colluding probabilities.

Trace preparation (2)

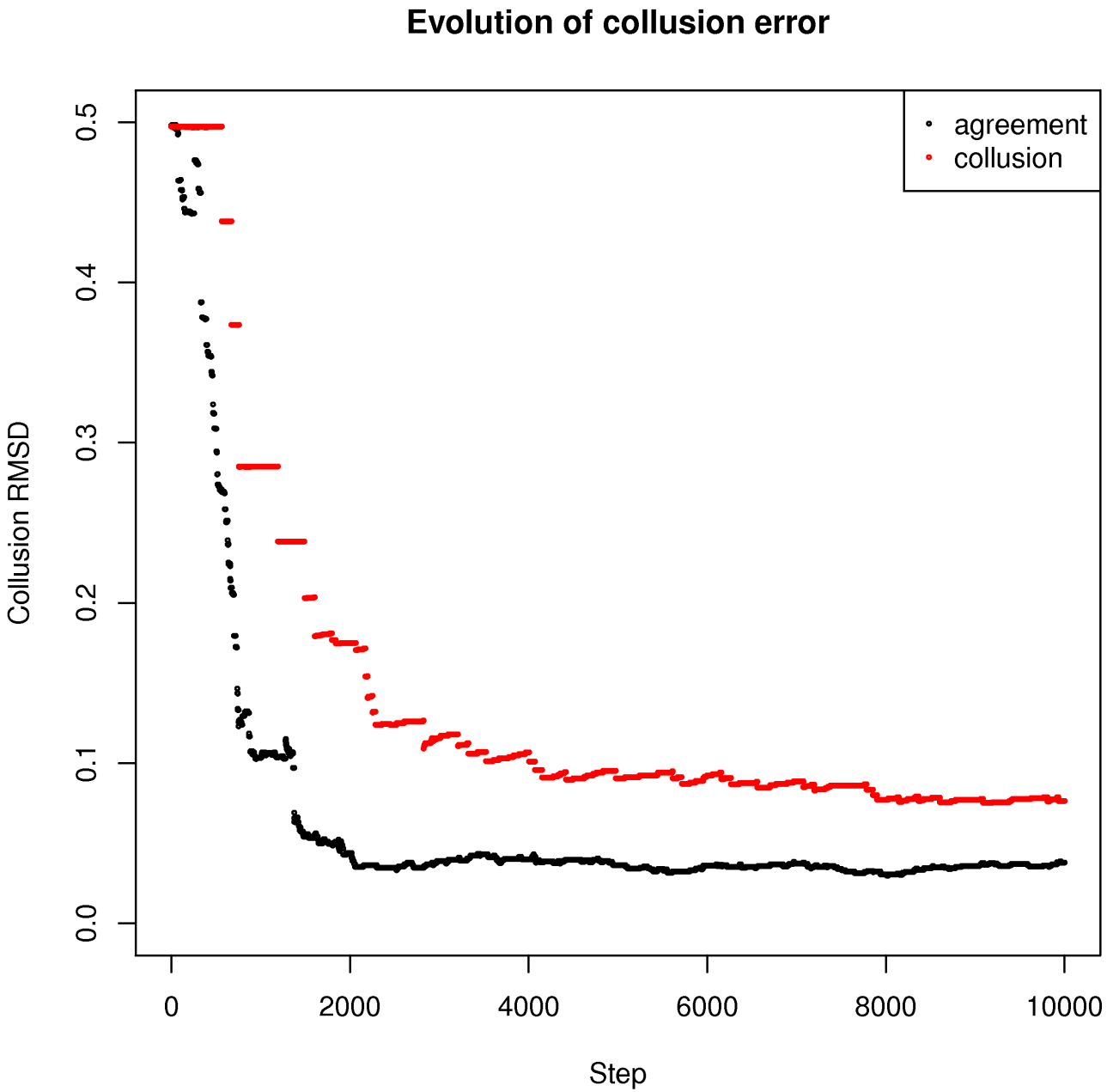
Scheduling

Redundancy-based scheduler: achieve a quorum of q with initial and maximal duplication l and l_{max}.
Jobs scheduled on workers when available (durations determined with the performance file and the normal duration of the job).
Result computations are based on reliability and colluding probabilities.

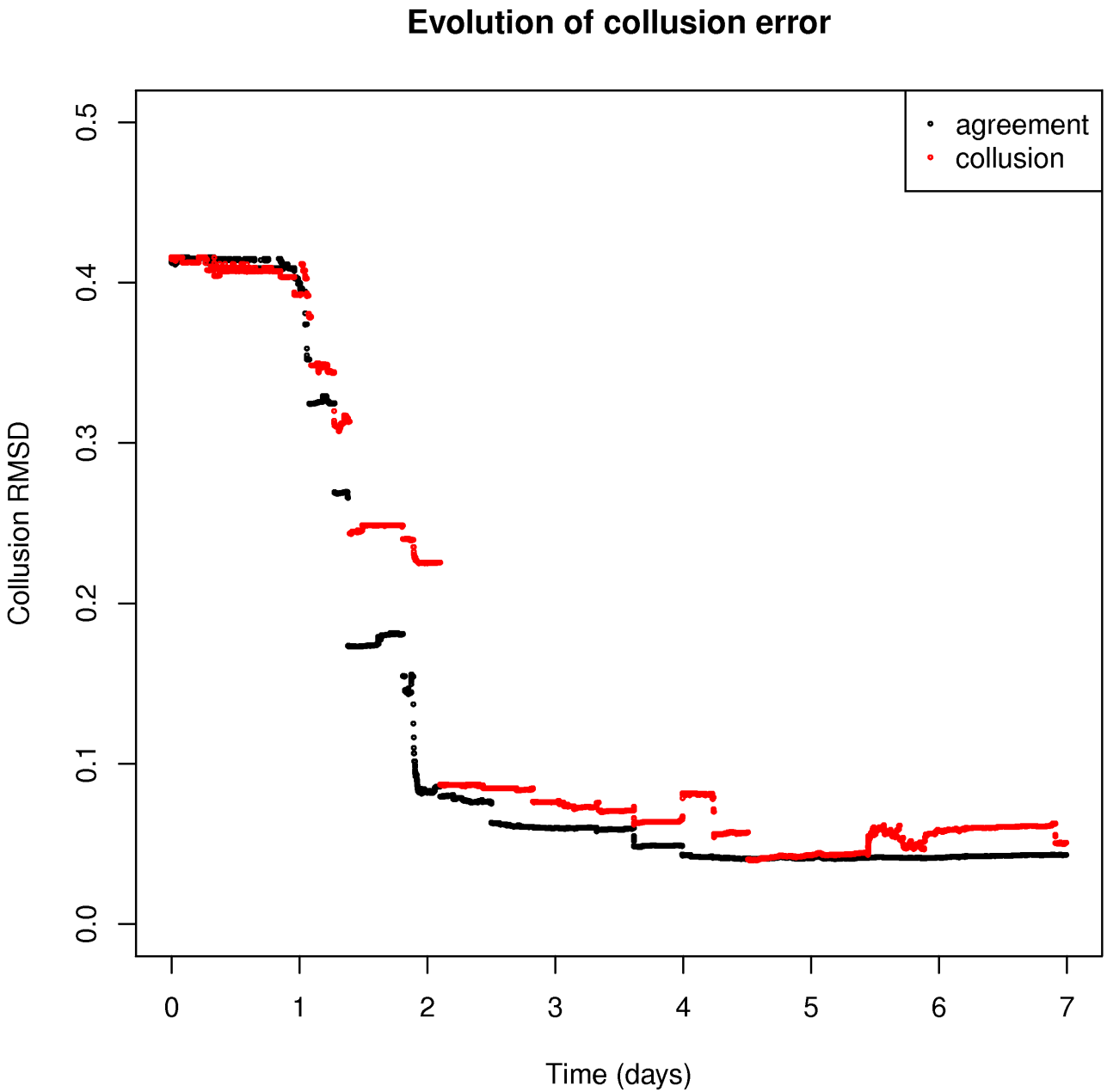
Trace that could have been useful

Job-centric traces on Desktop Grid: start and end times of the computation of a job; hash of the result.

Results with synthetic input



Results with traces



Outline

Conclusion and future directions

Main contributions

- Propose a reputation system
  - that observes interaction between groups of workers
  - with 2 representations (agreement, collusion)
- Use a realistic input based on existing traces

Perspective

- Correct the dependance reliability/collusion (the system does not converge)
- Apply clustering techniques based on agreement distance