A Dynamic Approach for Characterizing Collusion in Desktop Grids

Louis-Claude CANON, Emmanuel JEANNOT and Jon WEISSMAN

Project-Team Runtime
Labri/INRIA/Université Henri Poincaré
University of Minnesota

”Scheduling in Aussois” Workshop

June 2, 2010

Outline

1. Collusion in Desktop Grids
2. Characterization Mechanism
3. Empirical Validation
4. Conclusion

Outline

1 Collusion in Desktop Grids

2 Characterization Mechanism

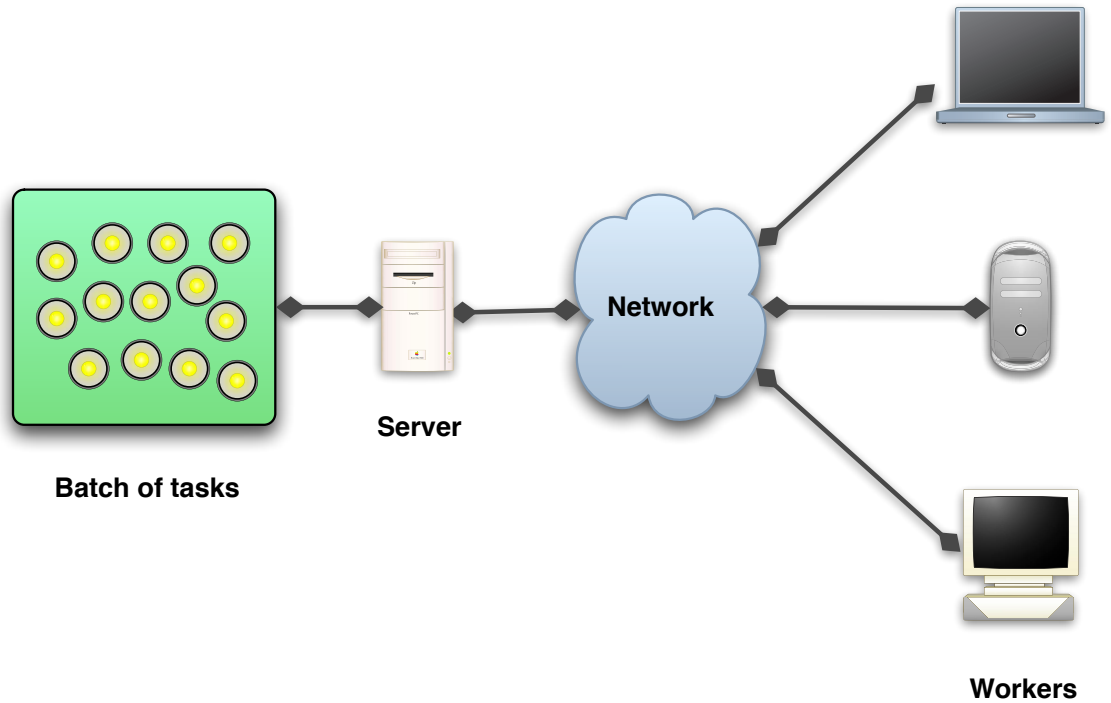
3 Empirical Validation

4 Conclusion

Desktop Grids

Distributed Computing

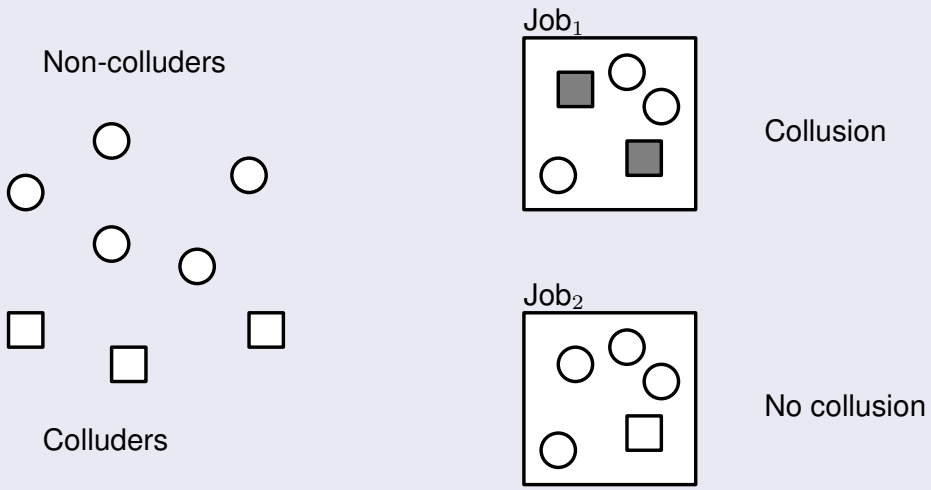
The server assigns **job** to each active **worker**.
Each worker returns a **result** for the corresponding job.
We want to have the *correct* result for each submitted job.
Example: BOINC-based projects.



Cheating

Collusion

Some workers produces the same wrong result for a given job.



Motivation

Causes of collusive behavior

Malicious participants.
Sybil attack (one entity for several identities).
Library with platform-specific bugs.

Objectives

Estimate collusion probabilities

Estimate the probability that any pair of workers gives the same wrong result for the same job.
Detecting the groups of workers that behave similarly.

Duplication

Constraints

Generic and non-intrusive mechanism: no information on the jobs.
No trustworthy computational resources: each result is unknown.

Duplication

Constraints

Generic and non-intrusive mechanism: no information on the jobs.
No trustworthy computational resources: each result is unknown.

Current BOINC solution

Assign a job to k distinct workers (redundancy) and select the result that has the majority.

Duplication

Constraints

Generic and non-intrusive mechanism: no information on the jobs.
No trustworthy computational resources: each result is unknown.

Current BOINC solution

Assign a job to k distinct workers (redundancy) and select the result that has the majority.

Extension proposal

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

Outline

1 Collusion in Desktop Grids

2 Characterization Mechanism

3 Empirical Validation

4 Conclusion

Interaction Model

Two interaction representations

Interaction between groups i and j:

collusion groups i and j collude together or not (collusion estimation c_{ij})

agreement groups i and j agree together or not (agreement estimation a_{ij})

Interaction Model

Two interaction representations

Interaction between groups i and j:

collusion groups i and j collude together or not (collusion estimation c_{ij})

agreement groups i and j agree together or not (agreement estimation a_{ij})

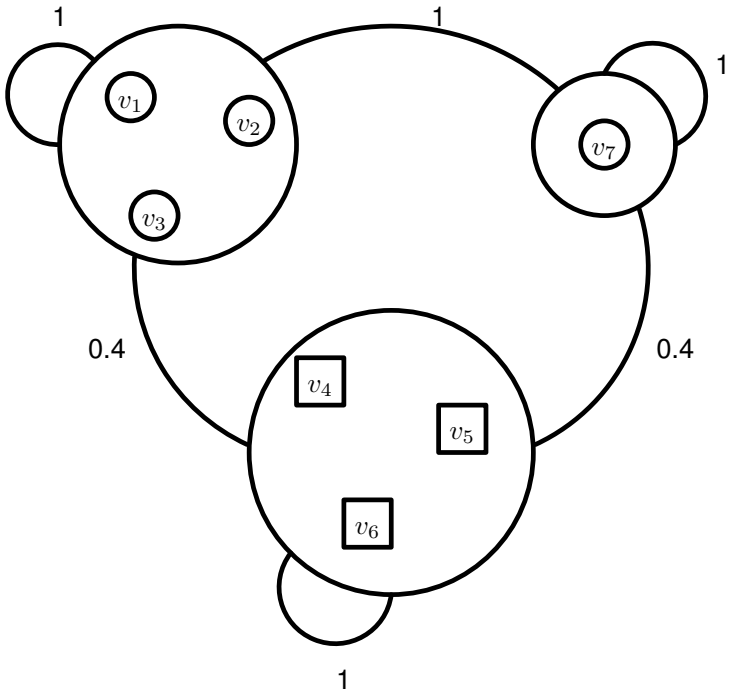
Relations

- $a_{ij} \leq 1 + 2 \times c_{ij} - c_{ii} - c_{jj}$
- $c_{ij} \leq \frac{1+a_{ij}-a_{1i}-a_{1j}}{2}$ (the index of the largest groups is 1)

On-line Algorithm

Data structure

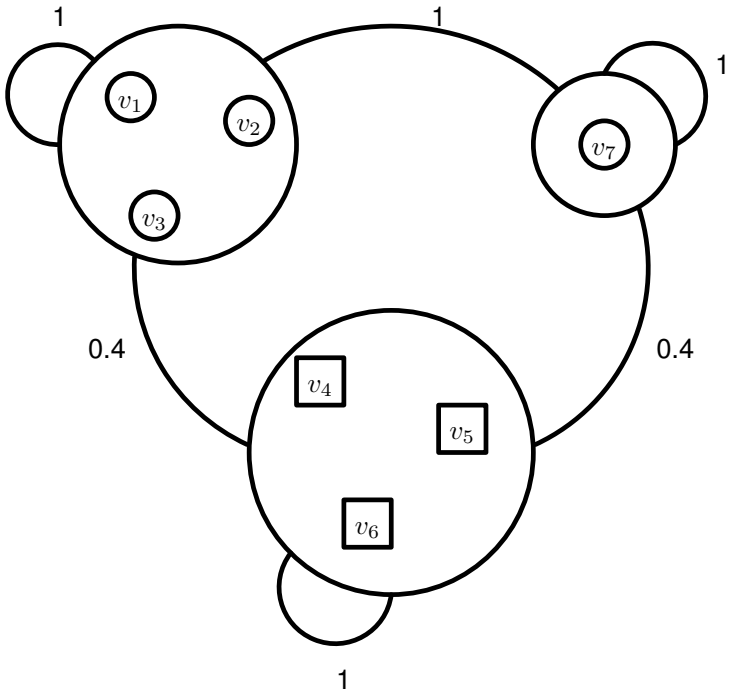
Graph: each node corresponds to a set of worker; each edge, to some collusion characteristics (agreement for the example on the right).



On-line Algorithm

Data structure

Graph: each node corresponds to a set of worker; each edge, to some collusion characteristics (agreement for the example on the right).



Algorithm

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

Agreement Criteria

Merge

Observed group i and j are merged if

- workers always returned the same result (*i.e.*, $a_{ij} \approx 1$)
- the number of observations is greater than $|i \cup j|$ (merge of small groups is easy while large groups need more observations)

Agreement Criteria

Merge

Observed group i and j are merged if

- workers always returned the same result (*i.e.*, $a_{ij} \approx 1$)
- the number of observations is greater than $|i \cup j|$ (merge of small groups is easy while large groups need more observations)

Split

If a worker disagree with any other from the same observed group: both conflicting workers are put in two new singletons

Agreement Criteria

Merge

Observed group i and j are merged if

- workers always returned the same result (*i.e.*, $a_{ij} \approx 1$)
- the number of observations is greater than $|i \cup j|$ (merge of small groups is easy while large groups need more observations)

Split

If a worker disagree with any other from the same observed group: both conflicting workers are put in two new singletons

Agreement *vs.* collusion

Agreement criteria are easier.

Outline

1 Collusion in Desktop Grids

2 Characterization Mechanism

3 Empirical Validation

4 Conclusion

Trace-Based Inputs

Process

Input: availibility traces (FTA), workload traces (GWA) and performance trace (FTA)
Output: collection of events ($< t, w, j, r >$ and $< t, j >$)

Trace-Based Inputs

Process

Input: availibility traces (FTA), workload traces (GWA) and performance trace (FTA)
Output: collection of events ($< t, w, j, r >$ and $< t, j >$)

Scheduling

Redundancy-based scheduler: achieve a quorum of q with initial and maximal duplication l and l_{max}.
Jobs scheduled on workers when available.
Result computations are based on reliability and colluding probabilities.
Mix real traces with scheduling and threat model for generating events.

Trace Parameters

Parameters

Parameter	Default value	Tested values
Worker availability trace	Seti@home	Overnet, Microsoft, ...
Workload model or trace	charmm	mfold, docking@home
Quorum (k, q, l)	(4, 3, 10)	(2, 1, 2), (19, 15, 19)
Number of workers (n)	100	30, 50, 70, 80, 200
Reliability (fraction, probability)	(0.7, 0.7)	$\{0.7, 0.99\} \times \{0.7, 0.99\} \setminus (0.7, 0.7)$
Collusion (fraction, probability)	(0.2, 0.5)	$\{0.02, 0.2, 0.49\} \times \{0.01, 0.5, 1\} \setminus (0.2, 0.5)$, Pair

Trace Parameters

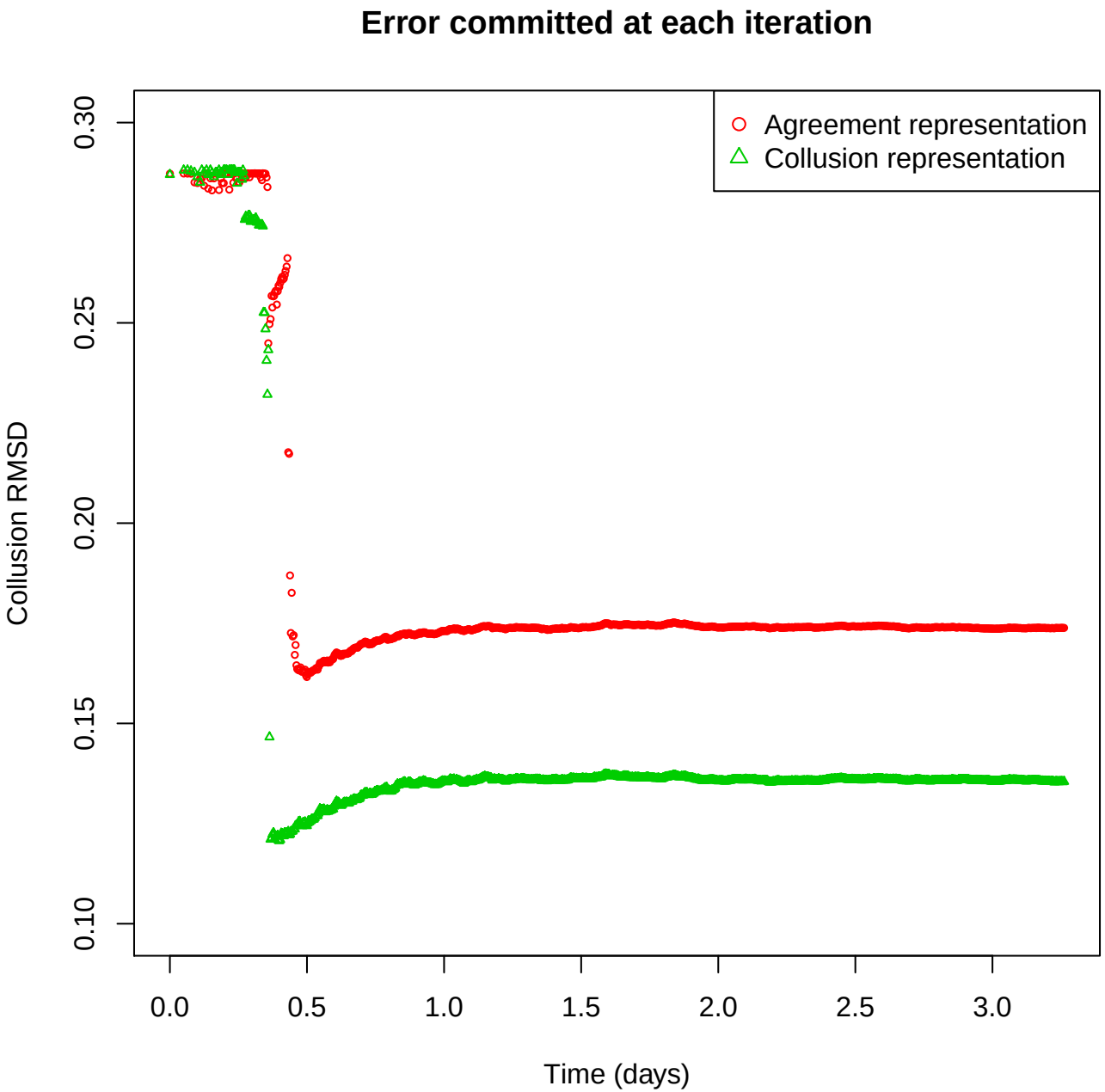
Parameters

Parameter	Default value	Tested values
Worker availability trace	Seti@home	Overnet, Microsoft, ...
Workload model or trace	charmm	mfold, docking@home
Quorum (k, q, l)	(4, 3, 10)	(2, 1, 2), (19, 15, 19)
Number of workers (n)	100	30, 50, 70, 80, 200
Reliability (fraction, probability)	(0.7, 0.7)	$\{0.7, 0.99\} \times \{0.7, 0.99\} \setminus (0.7, 0.7)$
Collusion (fraction, probability)	(0.2, 0.5)	$\{0.02, 0.2, 0.49\} \times \{0.01, 0.5, 1\} \setminus (0.2, 0.5)$, Pair

Summary

28 scenarios with 20 seeds: 560 traces on which both heuristics are run.

Specific Run



Outline

1 Collusion in Desktop Grids

2 Characterization Mechanism

3 Empirical Validation

4 Conclusion

Conclusion and future directions

Main contributions

- Propose a characterization system (with 2 representations)
- Validate with realistic inputs based on existing traces

Perspective

- Use both interaction models with the same group structure (agreement for updating structure, collusion to get the values)
- Use certification mechanism for fixing systematic error (single result may be correct)
- Higher-level scheduling mechanisms (oriented towards detection or avoidance)