Algorithmic Mechanisms for Internet-based Master-Worker Computing with Untrusted and Selfish Workers

Antonio Fernández Anta[1] Chryssis Georgiou[2] Miguel A. Mosteiro[1,3]

[1]LADyR, GSyC, Universidad Rey Juan Carlos

[2]Dept. of Computer Science, University of Cyprus

[3]Dept. of Computer Science, Rutgers University

IPDPS 2010

Motivation

- Demand for processing complex computational jobs
  - One-processor machines have limited computational resources
  - Powerful parallel machines are expensive
- Internet is emerging as an alternative platform for HPC
  - Volunteer computing: @home projects
    (e.g., SETI [Korpela et al 01])
  - Convergence of P2P and Grid computing
    [Foster, Iamnitchi 03]

Motivation

- Internet-based Computing
  - A Master machine acts as a server distributing jobs to client computers
  - Workers that execute and report back the results

(Internet-based Computing or P2P Computing - P2PC)

- Great potential
  - but limited use due to cheaters
    [Anderson 04; Golle, Mironov 01]
    cheater fabricates a bogus result and returns it

- Possible solution
  - redundant task-allocation
    [Anderson 04; Yurkewych et al 05; Fernández et al 06; etc.]
    1. the Master assigns same task to several workers and
    2. compares their returned results (voting)

Motivation

- Internet-based Computing
  - A Master machine acts as a server distributing jobs to client computers
  - Workers that execute and report back the results

(Internet-based Computing or P2P Computing - P2PC)

- Great potential
  - but limited use due to cheaters
    [Anderson 04; Golle, Mironov 01]
    cheater fabricates a bogus result and returns it

- Possible solution
  - redundant task-allocation
    [Anderson 04; Yurkewych et al 05; Fernández et al 06; etc.]
    1. the Master assigns same task to several workers and
    2. compares their returned results (voting)

Motivation

- Internet-based Computing
  - A Master machine acts as a server distributing jobs to client computers
  - Workers that execute and report back the results

  (Internet-based Computing or P2P Computing - P2PC)

- Great potential
  - but limited use due to cheaters
    [Anderson 04; Golle, Mironov 01]
    cheater fabricates a bogus result and returns it

- Possible solution
  - redundant task-allocation
    [Anderson 04; Yurkewych et al 05; Fernández et al 06; etc.]
    1. the Master assigns same task to several workers and
    2. compares their returned results (voting)

Motivation

Redundant task-allocation recent approaches

- "Classical" distributed computing (pre-defined worker behavior) [Fernández et al 06; Konwar et al 06]
  - malicious workers always report incorrect result (sw/hw errors, Byzantine, etc.)
  - altruistic workers always compute and truthfully report result (the "correct" nodes)

  Malicious-tolerant voting protocols are designed

- Game-theoretic (no pre-defined worker behavior) [Yurkewych et al 05; Babaioff et al 06; Fernández Anta et al 08]
  - rational workers act selfishly maximizing own benefit

  Incentives are provided to induce a desired behavior

- BUT realistically, the three types of workers may coexist!

Motivation

Redundant task-allocation recent approaches

- "Classical" distributed computing (pre-defined worker behavior)
  [Fernández et al 06; Konwar et al 06]
  - malicious workers always report incorrect result
    (sw/hw errors, Byzantine, etc.)
  - altruistic workers always compute and truthfully report result
    (the "correct" nodes)

  Malicious-tolerant voting protocols are designed

- Game-theoretic (no pre-defined worker behavior)
  [Yurkewych et al 05; Babaioff et al 06; Fernández Anta et al 08]
  - rational workers act selfishly maximizing own benefit

  Incentives are provided to induce a desired behavior

- BUT realistically, the three types of workers may coexist!

Motivation

Redundant task-allocation recent approaches

- "Classical" distributed computing (pre-defined worker behavior)
  [Fernández et al 06; Konwar et al 06]
  - malicious workers always report incorrect result
    (sw/hw errors, Byzantine, etc.)
  - altruistic workers always compute and truthfully report result
    (the "correct" nodes)

  Malicious-tolerant voting protocols are designed

- Game-theoretic (no pre-defined worker behavior)
  [Yurkewych et al 05; Babaioff et al 06; Fernández Anta et al 08]
  - rational workers act selfishly maximizing own benefit

  Incentives are provided to induce a desired behavior

- BUT realistically, the three types of workers may coexist!

Our approach

In this work: combine all

- Types of workers:
  - malicious: always report incorrect result
  - altruistic: always compute and report correct result
  - rational: selfishly choose to be honest or a cheater
- Game-theoretic approach:
  - Computations modeled as strategic games
  - Provide incentives to induce desired rationals behavior
- Classical distributed computing approach:
  - Design malice/altruism-aware voting games
  - Master chooses whether to audit the returned result or not
- Objective: reliable Internet-based computing
  - Minimize the probability of wrong result
  - Minimize master cost

Our approach

In this work: combine all

- Types of workers:
  - malicious: always report incorrect result
  - altruistic: always compute and report correct result
  - rational: selfishly choose to be honest or a cheater
- Game-theoretic approach:
  - Computations modeled as strategic games
  - Provide incentives to induce desired rationals behavior
- Classical distributed computing approach:
  - Design malice/altruism-aware voting games
  - Master chooses whether to audit the returned result or not
- Objective: reliable Internet-based computing
  - Minimize the probability of wrong result
  - Minimize master cost

Our approach

In this work: combine all

- Types of workers:
  - malicious: always report incorrect result
  - altruistic: always compute and report correct result
  - rational: selfishly choose to be honest or a cheater
- Game-theoretic approach:
  - Computations modeled as strategic games
  - Provide incentives to induce desired rationals behavior
- Classical distributed computing approach:
  - Design malice/altruism-aware voting games
  - Master chooses whether to audit the returned result or not
- Objective: reliable Internet-based computing
  - Minimize the probability of wrong result
  - Minimize master cost

Our approach

In this work: combine all

- Types of workers:
  - malicious: always report incorrect result
  - altruistic: always compute and report correct result
  - rational: selfishly choose to be honest or a cheater
- Game-theoretic approach:
  - Computations modeled as strategic games
  - Provide incentives to induce desired rationals behavior
- Classical distributed computing approach:
  - Design malice/altruism-aware voting games
  - Master chooses whether to audit the returned result or not
- Objective: reliable Internet-based computing
  - Minimize the probability of wrong result
  - Minimize master cost

Background

Definition

"A game consists of a set of players, a set of moves (or strategies) available to those players, and a specification of payoffs for each combination of strategies." [Wikipedia]

- Game Theory:
  - Players (processors) act on their self-interest
  - Rational [Golle, Mironov 01] behavior: seek to increase own utility choosing strategy according to payoffs
  - Protocol is given as a game
  - Design objective is to achieve equilibrium among players

Definition

Nash Equilibrium (NE): players do not increase their expected utility by changing strategy, if other players do not change [Nash 50]

Background

Definition

"A game consists of a set of players, a set of moves (or strategies) available to those players, and a specification of payoffs for each combination of strategies." [Wikipedia]

- Game Theory:
  - Players (processors) act on their self-interest
  - Rational [Golle, Mironov 01] behavior:
    seek to increase own utility choosing strategy according to payoffs
  - Protocol is given as a game
  - Design objective is to achieve equilibrium among players

Definition

Nash Equilibrium (NE): players do not increase their expected utility by changing strategy, if other players do not change [Nash 50]

Background

Definition

"A game consists of a set of players, a set of moves (or strategies) available to those players, and a specification of payoffs for each combination of strategies." [Wikipedia]

- Game Theory:
  - Players (processors) act on their self-interest
  - Rational [Golle, Mironov 01] behavior:
    seek to increase own utility choosing strategy according to payoffs
  - Protocol is given as a game
  - Design objective is to achieve equilibrium among players

Definition

Nash Equilibrium (NE): players do not increase their expected utility by changing strategy, if other players do not change [Nash 50]

Previous work

- Algorithmic Mechanism Design [Nisan, Ronen 01]
  Games designed to provide incentives s.t. players act "correctly"
  - Behave well: reward
  - Otherwise: penalize

  The design objective is to induce a desired behavior (e.g. unique NE)

- Game Theory in Distributed Computing [Halpern 07; Nisan et al 07]
  - Internet routing [Koutsoupias, Papadimitriou 99; Mavronicolas, Spirakis 01]
  - Resource location and sharing [Halldorsson et al 04]
  - Containment of Viruses spreading [Moscibroda et al 06]
  - Secret sharing [Halpern, Teague 04]
  - P2P services [Aiyer et al 05; Li et al 06 & 08]
  - Task allocation (only rationals)
    [Yurkewich et al 05; Babaioff et al 06; Fernández Anta et al 08]

Previous work

- Algorithmic Mechanism Design [Nisan, Ronen 01]
  Games designed to provide incentives s.t. players act “correctly”
  - Behave well: reward
  - Otherwise: penalize

  The design objective is to induce a desired behavior (e.g. unique NE)

- Game Theory in Distributed Computing [Halpern 07; Nisan et al 07]
  - Internet routing [Koutsoupias, Papadimitriou 99; Mavronicolas, Spirakis 01]
  - Resource location and sharing [Halldorsson et al 04]
  - Containment of Viruses spreading [Moscibroda et al 06]
  - Secret sharing [Halpern, Teague 04]
  - P2P services [Aiyer et al 05; Li et al 06 & 08]
  - Task allocation (only rationals)
    [Yurkewich et al 05; Babaioff et al 06; Fernández Anta et al 08]

Previous work

- Coexisting malicious and rational workers
  - k-fault tolerant NE [Eliaz 02]
    (Walrasian function computation)
  - BAR-tolerant protocol [Aiyer et al 02]
    (Cooperative backup service for P2P systems)
  - (k, t)-robust protocol (up to k rational colluders, t Byzantine workers)
    [Abraham et al 06]
    (Secret-sharing protocol)
  - BAR-tolerant gossip protocol [Li et al 06]
    (P2P live streaming application)
  - Malicious Bayesian games [Gairing 08]
    (Congestion control, distribution over malicious/rational)

Framework

- Master
  - Assigns a task to workers and collects responses
  - Can audit the values returned
    - Auditing may be cheaper that computing
    - The correct result might not be obtained
  - Goal: minimize master cost as long as $P_{wrong} \leq \varepsilon$
- Workers
  - Unknown type of workers $\longrightarrow$ Bayesian game [Harsanyi 1967]
  - Known probability distribution over types $(p_\rho + p_\mu + p_\alpha = 1)$
    - $p_\rho \longrightarrow$ Rational
    - $p_\mu \longrightarrow$ Malicious
    - $p_\alpha \longrightarrow$ Altruistic
  - All workers have to reply
  - Weak collusion (worst-case for voting): rationals decide independently, but all incorrect answers are the same
- Task
  - The probability of "guessing" the correct answer is negligible
  - The correct answer is unique

Framework

- Master
  - Assigns a task to workers and collects responses
  - Can audit the values returned
    - Auditing may be cheaper that computing
    - The correct result might not be obtained
  - Goal: minimize master cost as long as $P_{wrong} \leq \varepsilon$
- Workers
  - Unknown type of workers → Bayesian game [Harsanyi 1967]
  - Known probability distribution over types $(p_\rho + p_\mu + p_\alpha = 1)$
    - p_ρ → Rational
    - p_μ → Malicious
    - p_α → Altruistic
  - All workers have to reply
  - Weak collusion (worst-case for voting):
    rationals decide independently, but all incorrect answers are the same
- Task
  - The probability of "guessing" the correct answer is negligible
  - The correct answer is unique

Framework

- Master
  - Assigns a task to workers and collects responses
  - Can audit the values returned
    - Auditing may be cheaper that computing
    - The correct result might not be obtained
  - Goal: minimize master cost as long as $P_{wrong} \leq \varepsilon$
- Workers
  - Unknown type of workers → Bayesian game [Harsanyi 1967]
  - Known probability distribution over types ($p_\rho + p_\mu + p_\alpha = 1$)
    - p_ρ → Rational
    - p_μ → Malicious
    - p_α → Altruistic
  - All workers have to reply
  - Weak collusion (worst-case for voting):
    rationals decide independently, but all incorrect answers are the same
- Task
  - The probability of "guessing" the correct answer is negligible
  - The correct answer is unique

Contributions

General protocol

- Master assigns a task to n workers
- Rational worker cheats with probability $p_{\mathcal{C}}$ (seeking a NE)
- Master audits the responses with probability $p_{\mathcal{A}}$
- If master audits
  - rewards honest workers and
  - penalizes the cheaters
- If master does not audit
  - Accepts value returned by majority of workers
  - Rewards/penalizes according to one of four models

$\mathcal{R}_{\mathrm{m}}$	the master rewards the majority only
$\mathcal{R}_{\mathrm{a}}$	the master rewards all workers
$\mathcal{R}_{\emptyset}$	the master does not reward any worker
$\mathcal{R}_{\pm}$	the master rewards the majority and penalizes the minority

Note: reward models may be fixed exogenously or chosen by the master

Contributions

General protocol

- Master assigns a task to n workers
- Rational worker cheats with probability $p_{\mathcal{C}}$ (seeking a NE)
- Master audits the responses with probability $p_{\mathcal{A}}$
- If master audits
  - rewards honest workers and
  - penalizes the cheaters
- If master does not audit
  - Accepts value returned by majority of workers
  - Rewards/penalizes according to one of four models

$\mathcal{R}_{\mathrm{m}}$	the master rewards the majority only
$\mathcal{R}_{\mathrm{a}}$	the master rewards all workers
$\mathcal{R}_{\emptyset}$	the master does not reward any worker
$\mathcal{R}_{\pm}$	the master rewards the majority and penalizes the minority

Note: reward models may be fixed exogenously or chosen by the master

Contributions

General protocol

- Master assigns a task to n workers
- Rational worker cheats with probability $p_{\mathcal{C}}$ (seeking a NE)
- Master audits the responses with probability $p_{\mathcal{A}}$
- If master audits
  - rewards honest workers and
  - penalizes the cheaters
- If master does not audit
  - Accepts value returned by majority of workers
  - Rewards/penalizes according to one of four models

$\mathcal{R}_{\text{m}}$	the master rewards the majority only
$\mathcal{R}_{\text{a}}$	the master rewards all workers
$\mathcal{R}_{\emptyset}$	the master does not reward any worker
$\mathcal{R}_{\pm}$	the master rewards the majority and penalizes the minority

Note: reward models may be fixed exogenously or chosen by the master

Contributions

General protocol

- Master assigns a task to n workers
- Rational worker cheats with probability $p_{\mathcal{C}}$ (seeking a NE)
- Master audits the responses with probability $p_{\mathcal{A}}$
- If master audits
  - rewards honest workers and
  - penalizes the cheaters
- If master does not audit
  - Accepts value returned by majority of workers
  - Rewards/penalizes according to one of four models

$\mathcal{R}_{\mathrm{m}}$	the master rewards the majority only
$\mathcal{R}_{\mathrm{a}}$	the master rewards all workers
$\mathcal{R}_{\emptyset}$	the master does not reward any worker
$\mathcal{R}_{\pm}$	the master rewards the majority and penalizes the minority

Note: reward models may be fixed exogenously or chosen by the master

Contributions

Payoff parameters

$WP_{\mathcal{C}}$	worker's punishment for being caught cheating
$WC_{\mathcal{T}}$	worker's cost for computing the task
$WB_{\mathcal{Y}}$	worker's benefit from master's acceptance
$MP_{\mathcal{W}}$	master's punishment for accepting a wrong answer
$MC_{\mathcal{Y}}$	master's cost for accepting the worker's answer
$MC_{\mathcal{A}}$	master's cost for auditing worker's answers
$MB_{\mathcal{R}}$	master's benefit from accepting the right answer

Note: it is possible that $WB_{\mathcal{Y}} \neq MC_{\mathcal{Y}}$

Contributions

Characterize conditions for unique (mixed) NE
(under general type distribution for each reward model)

Design of mechanism to choose $p_{\mathcal{A}}$ parameterized on type-distribution
(minimize master cost as long as P_{wrong} is bounded by a parameter ε)

- It is shown that this mechanism is the only feasible approach to achieve a given bound on the probability of error.

Instantiate the mechanism in two real-world scenarios

- volunteering computing (SETI)
- contractor scenario
(company buys computing cycles from Internet computers and sells them to customers in the form of a task-computation service)

Contributions

Characterize conditions for unique (mixed) NE
(under general type distribution for each reward model)

Design of mechanism to choose $p_{\mathcal{A}}$ parameterized on type-distribution
(minimize master cost as long as P_{wrong} is bounded by a parameter ε)

- It is shown that this mechanism is the only feasible approach to achieve a given bound on the probability of error.

Instantiate the mechanism in two real-world scenarios

- volunteering computing (SETI)
- contractor scenario
(company buys computing cycles from Internet computers and sells them to customers in the form of a task-computation service)

Contributions

Characterize conditions for unique (mixed) NE
(under general type distribution for each reward model)

Design of mechanism to choose $p_{\mathcal{A}}$ parameterized on type-distribution
(minimize master cost as long as P_{wrong} is bounded by a parameter ε)

- It is shown that this mechanism is the only feasible approach to achieve a given bound on the probability of error.

Instantiate the mechanism in two real-world scenarios

- volunteering computing (SETI)
- contractor scenario
  (company buys computing cycles from Internet computers and sells them to customers in the form of a task-computation service)

Conditions for mixed-strategy NE (MSNE)

Definition

For a finite game, a mixed strategy profile σ^* is a MSNE iff, for each player i

$$U_i(s_i, \sigma_{-i}) = U_i(s'_i, \sigma_{-i}), \forall s_i, s'_i \in supp(\sigma_i)$$
$$U_i(s_i, \sigma_{-i}) \geq U_i(s'_i, \sigma_{-i}), \forall s_i, s'_i : s_i \in supp(\sigma_i), s'_i \notin supp(\sigma_i)$$

[Osborne 2003]

s_i : strategy of player i in strategy profile s

σ_i : probability distribution over pure strategies of player i in σ

$U_i(s_i, \sigma_{-i})$: expected utility of player i using strategy s_i in σ

$supp(\sigma_i)$: set of positive-probability strategies in σ

Conditions for mixed-strategy NE (MSNE)

Strategic payoffs

	$\mathcal{R}_\pm$	$\mathcal{R}_\mathrm{m}$	$\mathcal{R}_\mathrm{a}$	$\mathcal{R}_\emptyset$
$w_\mathcal{C}^\mathcal{A}$	$-WP_\mathcal{C}$	$-WP_\mathcal{C}$	$-WP_\mathcal{C}$	$-WP_\mathcal{C}$
$w_{\overline{\mathcal{C}}}^\mathcal{A}$	$WB_\mathcal{Y} - WC_\mathcal{T}$	$WB_\mathcal{Y} - WC_\mathcal{T}$	$WB_\mathcal{Y} - WC_\mathcal{T}$	$WB_\mathcal{Y} - WC_\mathcal{T}$
$w_\mathcal{C}^\mathcal{C}$	$WB_\mathcal{Y}$	$WB_\mathcal{Y}$	$WB_\mathcal{Y}$	0
$w_{\overline{\mathcal{C}}}^\mathcal{C}$	$-WP_\mathcal{C} - WC_\mathcal{T}$	$-WC_\mathcal{T}$	$WB_\mathcal{Y} - WC_\mathcal{T}$	$-WC_\mathcal{T}$
$w_\mathcal{C}^{\overline{\mathcal{C}}}$	$-WP_\mathcal{C}$	0	$WB_\mathcal{Y}$	0
$w_{\overline{\mathcal{C}}}^{\overline{\mathcal{C}}}$	$WB_\mathcal{Y} - WC_\mathcal{T}$	$WB_\mathcal{Y} - WC_\mathcal{T}$	$WB_\mathcal{Y} - WC_\mathcal{T}$	$-WC_\mathcal{T}$

$w_{s_i}^\mathcal{X}$ payoff of player i using strategy $s_i \in \{\mathcal{C}, \overline{\mathcal{C}}\}$ if

$$\mathcal{X} = \begin{cases} \mathcal{A} & \text{master audits} \\ \mathcal{C} & \text{majority of workers cheat and master does not audit} \\ \overline{\mathcal{C}} & \text{majority of workers does not cheat and master does not audit} \end{cases}$$

Conditions for mixed-strategy NE (MSNE)

For each player i and each reward model, enforce unique NE in

$$\Delta U = U_i(s_i = \mathcal{C}, \sigma_{-i}) - U_i(s_i = \overline{\mathcal{C}}, \sigma_{-i})$$

$$\Delta U = (w_{\mathcal{C}}^{\mathcal{A}} - w_{\overline{\mathcal{C}}}^{\mathcal{A}})p_{\mathcal{A}} + (1 - p_{\mathcal{A}})\Bigg((w_{\mathcal{C}}^{\mathcal{C}} - w_{\overline{\mathcal{C}}}^{\mathcal{C}})\mathbf{P}_q^{(n-1)}(\lceil n/2 \rceil, n-1) + (w_{\mathcal{C}}^{\overline{\mathcal{C}}} - w_{\overline{\mathcal{C}}}^{\overline{\mathcal{C}}})\mathbf{P}_q^{(n-1)}(0, \lfloor n/2 \rfloor - 1) + (w_{\mathcal{C}}^{\mathcal{C}} - w_{\overline{\mathcal{C}}}^{\overline{\mathcal{C}}})\binom{n-1}{\lfloor n/2 \rfloor} q^{\lfloor n/2 \rfloor}(1-q)^{\lfloor n/2 \rfloor}\Bigg)$$

where $q = p_\mu + p_\rho p_{\mathcal{C}}$, $\mathbf{P}_q^{(n)}(a,b) = \sum_{i=a}^{b} \binom{n}{i} q^i (1-q)^{n-i}$

Computational issues: together with the task, the master sends a “certificate” ($p_{\mathcal{A}}$, payoffs, n) of the uniqueness of the desired NE to the worker

Conditions for mixed-strategy NE (MSNE)

ensuring

$$P_{wrong} \leq \varepsilon$$

while maximizing

$$\max U_M$$

$$\begin{aligned} P_{wrong} &= (1 - p_{\mathcal{A}})\mathbf{P}_q^{(n)}(\lceil n/2 \rceil, n) \\ U_M &= p_{\mathcal{A}}\big(MB_{\mathcal{R}} - MC_{\mathcal{A}} - n(1-q)MC_{\mathcal{Y}}\big) + \\ &\quad (1 - p_{\mathcal{A}})\big(MB_{\mathcal{R}}\mathbf{P}_q^{(n)}(0, \lfloor n/2 \rfloor) - MP_{\mathcal{W}}\mathbf{P}_q^{(n)}(\lceil n/2 \rceil, n) + \gamma\big) \end{aligned}$$

where

$$\gamma = \begin{cases} -MC_{\mathcal{Y}}(\mathbf{E}_{1-q}^{(n)}(\lceil n/2 \rceil, n) + \mathbf{E}_q^{(n)}(\lceil n/2 \rceil, n)) & \mathcal{R}_{\mathrm{m}} \text{ and } \mathcal{R}_{\pm} \text{ models} \\ -nMC_{\mathcal{Y}} & \mathcal{R}_{\mathrm{a}} \text{ model} \\ 0 & \mathcal{R}_{\emptyset} \text{ model} \end{cases}$$

$$\mathbf{E}_p^{(n)}(a, b) = \sum_{i=a}^{b} \binom{n}{i} i p^i (1-p)^{n-i}, p \in [0, 1]$$

Mechanism design

Master protocol to choose $p_{\mathcal{A}}$

- case even if $p_{\mathcal{C}} = 0$, P_{wrong} is big $(\mathbf{P}^{(n)}_{p_\mu}(\lceil n/2 \rceil, n) > \varepsilon)$

  $$p_{\mathcal{A}} \leftarrow 1 - \varepsilon / \mathbf{P}^{(n)}_{p_\mu + p_\rho}(\lceil n/2 \rceil, n)$$

- case even if $p_{\mathcal{C}} = 1$, P_{wrong} is low $(\mathbf{P}^{(n)}_{p_\mu + p_\rho}(\lceil n/2 \rceil, n) \leq \varepsilon)$

  $$p_{\mathcal{A}} \leftarrow 0$$

- case $p_{\mathcal{C}} = 0$, even if $p_{\mathcal{A}} = 0$ $(\Delta U(p_{\mathcal{C}} = 1, p_{\mathcal{A}} = 0) \leq 0$ and $(\mathcal{R}_{\mathrm{m}} \vee \mathcal{R}_{\pm}))$

  $$p_{\mathcal{A}} \leftarrow 0$$

- otherwise $p_{\mathcal{C}} = 0$ enforced

$$p_{\mathcal{A}} \leftarrow \begin{cases} 1 - \dfrac{WP_{\mathcal{C}} + WB_{\mathcal{Y}} - WC_{\mathcal{T}}}{WP_{\mathcal{C}} + WB_{\mathcal{Y}}(\mathbf{P}^{(n-1)}_{p_\mu + p_\rho}(\lfloor n/2 \rfloor, n-1) + \mathbf{P}^{(n-1)}_{p_\mu + p_\rho}(\lceil n/2 \rceil, n-1))} & \mathcal{R}_{\mathrm{m}} \\ \dfrac{WC_{\mathcal{T}}}{WP_{\mathcal{C}} + WB_{\mathcal{Y}}} + \psi, \text{ for any } \psi > 0 & \mathcal{R}_{\mathrm{a}} \ \&\ \mathcal{R}_{\emptyset} \\ 1 - \dfrac{WP_{\mathcal{C}} + WB_{\mathcal{Y}} - WC_{\mathcal{T}}}{(WP_{\mathcal{C}} + WB_{\mathcal{Y}})(\mathbf{P}^{(n-1)}_{p_\mu + p_\rho}(\lfloor n/2 \rfloor, n-1) + \mathbf{P}^{(n-1)}_{p_\mu + p_\rho}(\lceil n/2 \rceil, n-1))} & \mathcal{R}_{\pm} \end{cases}$$

- if $U_M(p_{\mathcal{A}}, q) < U_M(1 - \varepsilon, p_\mu + p_\rho)$ then $p_{\mathcal{A}} \leftarrow 1 - \varepsilon$

Mechanism design

Optimality

Only feasible approach for $P_{wrong} \leq \varepsilon$

Theorem

In order to achieve $P_{wrong} \leq \varepsilon$, the only feasible approaches are either to enforce a NE where $p_{\mathcal{C}} = 0$ or to choose $p_{\mathcal{A}}$ so that $P_{wrong} \leq \varepsilon$ even if all rationals cheat.

Proof.

ΔU is increasing in q $(\Delta U(p_{\mathcal{C}} < 1) \leq \Delta U(p_{\mathcal{C}} = 1))$

$\rightarrow$ the only unique NE corresponds to $p_{\mathcal{C}} = 0$.

For any other NE where $p_{\mathcal{C}} > 0$, $p_{\mathcal{C}} = 1$ is also a NE

$\rightarrow$ P_{wrong} worst case when all players cheat. □

Real-world scenarios

Volunteering computing (SETI-like)

- each worker
  - incurs in no cost to perform the task ($WC_{\mathcal{T}} = 0$)
  - obtains a benefit ($WB_{\mathcal{Y}} > 0$) (recognitiion, prestige)
- master
  - incurs in a (possibly small) cost to reward a worker ($MC_{\mathcal{Y}} > 0$) (advertise participation)
  - may audit results at a cost ($MC_{\mathcal{A}} > 0$)
  - obtains a benefit for correct result ($MB_{\mathcal{R}} > MC_{\mathcal{Y}}$)
  - suffers a cost for wrong result ($MP_{\mathcal{W}} > MC_{\mathcal{A}}$)

Instantiating the mechanism designed on these conditions the master can choose $p_{\mathcal{A}}$ and n so that U_M is maximized for $P_{wrong} \leq \varepsilon$ for any given worker-type distribution, reward model, and set of payoff parameters in the SETI scenario.

Real-world scenarios

Contractor scenario

- master
  - pays each worker an amount ($MC_{\mathcal{Y}} > 0$)
  - receives a benefit (from consumers for the provided service) ($MB_{\mathcal{R}} > MC_{\mathcal{Y}}$)
  - may audit and has a cost for wrong result ($MP_{\mathcal{W}} > MC_{\mathcal{A}} > 0$)
- each worker
  - receives payment for computing the task (not volunteers) ($WB_{\mathcal{Y}} = MC_{\mathcal{Y}}$)
  - incurs in a cost for computing ($WC_{\mathcal{T}} > 0$)
  - must have economic incentive ($U > 0$)

Instantiating the mechanism designed on these conditions the master can choose $p_{\mathcal{A}}$ and n so that U_M is maximized for $P_{wrong} \leq \varepsilon$ for any given worker-type distribution, reward model, and set of payoff parameters in the contractor scenario.

Conclusions

Summary

- combination of approaches
  - classical distributed computing (voting)
  - game-theoretic (cost-based incentives and payoffs)
- algorithm to reliably obtain a task result despite the co-existence of malicious, altruistic and rational workers.
- mechanism to trade reliability (ε) and cost (U_M)
- as an example: instantiation of such algorithm in two real-world scenarios
- BOINC-based systems (such as SETI@home) send the same task to three (3) workers. Our analysis identifies rigorously, for any given system parameters, the best allocation that BOINC-based systems could deploy.
- the analysis on the contractor scenario opens the way for commercial Internet-based supercomputing where a company, given specific system parameters, could calculate its profit (if any) before agreeing into providing a proposed computational service.

Future work

- more involved collusion (beyond returning the same incorrect result)
- unreliable network (some replies do not arrive)
- multiple rounds protocol (worker reputation)

Thank you