Brief Announcement: Achieving Reliability in Master-Worker Computing via Evolutionary Dynamics

Evgenia Christoforou
University of Cyprus
christoforou.evgenia@ucy.ac.cy

Antonio Fernández Anta
Institute IMDEA Networks
& Univ. Rey Juan Calros
antonio.fernandez@imdea.org

Chryssis Georgiou
University of Cyprus
chryssis@ucy.ac.cy

Miguel A. Mosteiro
Rutgers University
& Univ. Rey Juan Carlos
mosteiro@cs.rutgers.edu

Angel (Anxo) Sánchez
Univ. Carlos III de Madrid
& BIFI Institute
anxo@math.uc3m.es

ABSTRACT

This work considers Internet-based task computations in which a master process assigns tasks, over the Internet, to rational workers and collect their responses. The objective is for the master to obtain the correct task outcomes. For this purpose we formulate and study the dynamics of evolution of Internet-based master-worker computations through reinforcement learning.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems

General Terms

Reliability, Algorithms

Keywords

Internet-based task computing, Evolutionary dynamics, Reinforcement learning, Algorithmic mechanism design.

1. INTRODUCTION

Motivation and prior work: As an alternative to expensive supercomputing parallel machines, Internet is a feasible computational platform for processing complex computational jobs. Several Internet-based applications operate on top of this global computation infrastructure. Examples are the volunteer-based "@home" projects such as SETI. In SETI, for example, there is a machine, call it the *master*, that sends tasks, across the Internet, to volunteers' computers, call them *workers*. These workers execute and report back some result. However, these workers may not be trustworthy (limiting the platforms potentials) and it might be at their best interest to report incorrect results; that is, workers, or their owners, can be viewed as *rational* [2]. In SETI, the master attempts to minimize the impact of these bogus results by assigning the same task to several workers and comparing their outcomes (i.e., redundant task allocation is employed).

Prior work has shown that it is possible to design algorithmic mechanisms with reward/punish schemes so that the master can reliably obtain correct task results. We view these mechanisms as one-shot in the following sense: In a round, the master sends a task to be computed to a collection of workers, and the mechanism, using auditing and reward/punish schemes guarantees (with high probability) that the master gets the correct task result. For another task to be computed, the process is repeated (with the same or different collection of workers) but without taking advantage of the knowledge gained. *A detailed account of related work can be found in [4].*

Given a long running computation (such as SETI-like master-worker computations), it can be the case that the best interests, and hence the behavior of the workers, might change over time. So, one wonders: Would it be possible to design a mechanism for performing many tasks, over the course of a possibly infinite computation, that could positively exploit the repeated interaction between a master and the same collection of workers?

Our approach: In this work we provide a positive answer to the above question. To do so, we introduce the concept of *evolutionary dynamics* under the biological and social perspective and relate them to Internet-based master-worker task computing. More specifically, we employ *reinforcement learning* [3] to model how system entities or learners interact with the environment to decide upon a strategy, and use their experience to select or avoid actions according to the consequences observed. Positive payoffs increase the probability of the strategy just chosen, and negative payoffs reduce this probability. Payoffs are seen as parameterizations of players' responses to their experiences. Empirical evidence [1] suggests that reinforcement learning is more plausible with players that have information only on the payoffs they receive; they do not have knowledge of the strategies involved. This model of learning fits nicely to our master-worker computation problem: the workers have no information about the master and the other workers' strategies and they don't know the set of strategies that led to the payoff they receive. The workers have only information about the strategies they choose at each round of the evolution of the system and their own received payoffs. The master also has minimal information about the workers and their intentions (to be truthful or not). Thus, we employ reinforcement learning for both the master and the workers in an attempt to build a reliable computational platform.

PODC'12, July 16–18, 2012, Madeira, Portugal.
ACM 978-1-4503-1450-3/12/07.

Our contributions: (Full details in [4].)

- We formulate and study the dynamics of the evolution of Internet-based master-worker computations through reinforcement learning.
- We develop and analyze a mechanism based on reinforcement learning to be used by the master and the workers. In particular, in each round, the master allocates a task to the workers and decides whether to audit or not their responses with a certain probability $p_{\mathcal{A}}$. Depending on whether it audits or not, it applies a different reward/punish scheme, and adjusts the probability $p_{\mathcal{A}}$ for the next round (a.k.a. the next task execution). Similarly, in a round, each worker i decides whether it will truthfully compute and report the correct task result, or it will report an incorrect result, with a certain probability p_{Ci}. Depending on the success or not of its decision, measured by the increase or the decrease of the worker's *utility*, the worker adjusts probability p_{Ci} for the next round.
- We show necessary and sufficient conditions under which the mechanism ensures *eventual correctness*, that is, we show the conditions under which, after some finite number of rounds, the master obtains the correct task result in every round, with minimal auditing, while keeping the workers satisfied (w.r.t. their utility).
- Finally, we show that our mechanism, when adhering to the above-mentioned conditions, reaches eventual correctness quickly. In particular, we show analytically, probabilistic bounds on the convergence time, as well as bounds on the expected convergence time. Our analysis is complemented with simulations.

2. ALGORITHMIC MECHANISM

The mechanism is composed by an algorithm run by the Master and an algorithm run by each worker.

Master's Algorithm: At each round, the master sends a task to all workers in W ($|W| = n$) and, when all answers are received, the master audits the answers with probability $p_{\mathcal{A}}$; auditing means that the master computes the task by itself, and checks which workers have truthfully reported the correct task result. We assume that there is a value $p_{\mathcal{A}}^{min} > 0$ so that at all times $p_{\mathcal{A}} \geq p_{\mathcal{A}}^{min}$. In the case the answers are not audited, the master accepts the value contained in the majority of answers and continues to the next round with the same probability of auditing. In the case the answers are audited, the value $p_{\mathcal{A}}$ of the next round is reinforced; meaning that $p_{\mathcal{A}}$ is modified according to the outcome of the round as follows:

$$p_{\mathcal{A}} = \min\{1, \max\{p_{\mathcal{A}}^{min}, p_{\mathcal{A}} + \alpha_m(\tfrac{cheaters(r)}{n} - \tau)\}\}.$$

The master initially has scarce or no information about the environment (e.g., workers initial p_C). Therefore, a safe approach for the master is to initially set $p_{\mathcal{A}} = 0.5$. Observe that, in a round r, when the answers are not audited, the master has no information about the number of cheaters $cheaters(r)$. Thus, $p_{\mathcal{A}}$ remains the same as in the previous round. When the answers are audited, the master can safely extract $cheaters(r)$ and the master adapts the auditing probability $p_{\mathcal{A}}$ accordingly.

A discount factor, which we call *tolerance* and denote by τ, expresses the master's tolerable ratio of cheaters (typically, we will assume $\tau = 1/2$). Hence, if the proportion of cheaters is larger than τ, $p_{\mathcal{A}}$ will be increased, and otherwise, $p_{\mathcal{A}}$ will be decreased. The amount by which $p_{\mathcal{A}}$ changes depends on the difference between these values, modulated by a *learning rate* α_m. This latter value determines to what extent the newly acquired information will override the old information.

After the master has received all answers , rewards/penalizes the workers appropriately. The following workers' payoff parameters are considered: (1) $WP_{\mathcal{C}}$: worker's punishment for being caught cheating; (2) $WC_{\mathcal{T}}$: worker's cost for computing the task; (3) $WB_{\mathcal{Y}}$: worker's benefit from master's acceptance. Furthermore, in every round, a worker i has an *aspiration* a_i, that is, the minimum benefit it expects to obtain in a round. To motivate the workers to participate in the computation, the master must ensure that $WB_{\mathcal{Y}} \geq a_i$.We assume that the master knows the aspirations (it may be included in a contract the master and the worker agree before the start of the computation).

Workers' Algorithm: At each round, each worker receives a task from the master and, with probability $1 - p_{Ci}$ calculates the task, and replies to the master with the correct answer. (Initially, p_{Ci} could be set to 0.5.) If the worker decides to cheat, it fabricates an answer, and sends the incorrect response to the master. Flag S_i models the decision of a worker i to cheat ($S_i = -1$) or not ($S_i = 1$). After receiving its payoff, each worker i changes its p_{Ci} according to the payoff Π_i received, the chosen strategy S_i, and its aspiration a_i as follows:

$$p_{Ci} = \max\{0, \min\{1, p_{Ci} - \alpha_w(\Pi_i - a_i)S_i\}\}.$$

The workers have a learning rate α_w. We assume that all workers have the same learning rate, that is, they learn in the same manner (see also the discussion in [3]; the learning rate is called step-size there); note that our analysis can be adjusted to accommodate also workers with different learning rates.

Overview of results: We analyze the evolution of the master-worker system as a Markov chain and we show the following result:

THEOREM 1. *If $p_{\mathcal{A}} > 0$ then, in order to guarantee with positive probability that, after some finite number of rounds, the system achieves eventual correctness, it is* **necessary** *and* **sufficient** *to set $WB_{\mathcal{Y}} \geq a_i + WC_{\mathcal{T}}$ for all $i \in Z$ in some set $Z \subseteq W$ such that $|Z| > n/2$.*

We call the time (number of rounds) taken to achieve eventual convergence as *convergence time*. We show, both in expectation and with high probability, that when our mechanism adheres to the conditions of Theorem 1, it can reach convergence time quickly. Our analysis is complemented with simulation results that further demonstrate the practicality of our mechanism. Full details can be found in [4].

Acknowledgments: This work is supported by the Cyprus Research Promotion Foundation Grant ΤΠΕ/ΠΛΗΡΟ/0609(ΒΕ)/05.

3. REFERENCES

[1] C. F. Camerer. Behavioral game theory: Experiments in strategic interaction. *Roundtable Series in Behavioral Economics*, 2003.

[2] J. Shneidman and D.C. Parkes. Rationality and self-interest in P2P networks. In *IPTPS 2003*, pp. 139–148.

[3] C. Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan & Claypool publishers, 2010.

[4] Technical report of this work: `http://www.cs.ucy.ac.cy/~chryssis/EvolMW-TR.pdf`