# Classification-Based Reputation Mechanism for Master-Worker Computing System

Kun Lu(✉), Jingchao Yang, Haoran Gong, and Mingchu Li

School of Software Technology, Dalian University of Technology, Dalian 116620, China
lukun@dlut.edu.cn

**Abstract.** Master-worker computing is a parallel computing scheme, which makes master and worker collaborate. Due to its high reliability availability and serviceability, it is widely used in scientific computing fields. However, lack of cooperation and malicious attack in Master-worker computing can greatly reduce the efficiency of parallel computing. In this paper, we consider a reputation system based on individual classification to inducing worker nodes returning true answer and separate malicious worker nodes. By introducing reinforcement learning, rational workers are induced to behave cooperatively and auditing rate of the master decreases. Our model is based on evolutionary game theory. Simulation results show that our reputation system can not only effectively guarantee eventual correctness, separate malicious worker nodes, but also save the master node's auditing cost.

**Keywords:** Node classification · Reinforcement learning · Reputation system

## 1 Introduction

The high-performance computing is needed in scientific field over past decades. Many internet-based systems are proposed over years, such as SETI [1], Turk [2], etc. Master-worker model (MW-model) [3] is a widely used high-performance computing model. In MW-model, there is one master node and several worker nodes. In each computing task, the master node first sends the task to all worker nodes; then worker nodes return a computing result; finally, master then evaluates each worker's results. Each worker node that helps computer the task may get a reward.

However, computing a task cost a lot of resources of each worker node, such as CPU and memory. Due to the rational nature, the worker nodes tend to re- turn random results without actual computation. Thus, incentivizing rational workers to perform cooperatively in distributed systems is a critical issue [4, 5].

Reward and punishment mechanisms [6–8] are most widely used mechanisms in promoting cooperation in MW-model. Generally, in MW-model, those cooperative worker nodes that return correct results are given extra rewards and those worker nodes

This paper is supported by the Liaoning Provincial National Science Foundation of China under grant No. 2017540158.


L. Wang et al. (Eds.): QShine 2017, LNICST 234, pp. 238–247, 2018.
https://doi.org/10.1007/978-3-319-78078-8_24

return false results may be sanctioned a penalty. As mentioned before, worker nodes are treated as rational and strategic workers, game theory is an appropriate tool to model the nodes and interactions among them [9].

Preventing malicious attacks is a critical issue in MW-model. As worker nodes are rational, they tend to return false results. To solve this problem, Kondo et al. [15] proposed a classical solution to consider all workers are altruistic and proposed a malicious-tolerant protocol. However, malicious nodes intentionally send false results with complex behavior modes. A most common attack is persistent attack that a malicious node keeps sending false results to master node. One of the most dangerous attack is on-off attack, where malicious nodes send good and dangerous services in turn to keep reputation to a certain level. Only be tolerant to malicious attacks is not enough to guarantee system robustness. Reputation management systems are proved to be most effective to resist attacks. Generally, a reputation system relies on users' feedbacks: a worker node that provides a positive feedback. A worker node with a high reputation has a higher probability to be chosen as a correct result and to be rewarded. However, the effectiveness of getting accurate reputation is critical.

In this paper, we propose a classification based reputation system for master-worker computing scheme. Workers are divided into two types: type A and type B. Type A workers are those can be fully trusted and the master node takes their returned results as correct results in priority. And type B workers are those worker nodes with reputation less than 1. Type B workers can be promoted to Type A only if they continuously send correct results. Vice versa, Type A worker can be degrade to Type B if they send false results. Both worker and master nodes use enforcement learning to adapt cheating rate and audit rate. Simulation results show that, with our proposed mechanism, the system can quickly evolve to system eventual correctness and save master node's cost on auditing.

The rest of this paper is organized as follows. In Sect. 2, we introduce our system model in details. In Sect. 3, we present system evaluation results and analysis. In Sect. 4, we conclude this paper.

## 2 System Model

### 2.1 System Overview

In this paper, we consider a static internet-based master-worker system, where no worker nodes join and leave the system after initialization.

Consider that one master node distributes tasks to a set of $W$ with $n$ worker nodes. A subset $Wc$ of workers return correct answer and a subset $W_f$ of workers return false answer, where $W_c \subseteq W$ and $W_f = W \backslash W_c$. The computation of a task consists of multiple rounds. In each round, the master node sends a subtask to all worker nodes and the worker nodes return the result. Then, the master node evaluates the result, and reward worker nodes with correct answer and punish those with false answer.

The basic assumptions are as followings: (1) each subtask has one unique right answer and all false answers are same; (2) each worker node finishes a task individually;

(3) the goal of the master node is to get eventual correctness; (4) both the master node and worker nodes are rational, they participate in the system to maximize their own payoffs.

## 2.2 Game-Based Transaction Model

In our model, both master and worker nodes are rational and selfish, they are trying to maximize their benefits. Thus, Game theory is a suitable tool to model these nodes and interactions among them.

Computing is costly to the worker nodes. Thus, for each computing task, the cooperative worker node bears a $WC_t$ (see Table 1) cost if it really does the computing. To incentive rational worker compute the tasks, the master node implements auditing, reward and punishment mechanism.

**Table 1.** Definition of notations

| Notation | Definition |
|---|---|
| $WB_y$ | reward of workers returning correct answer |
| $WC_t$ | cost of a worker computing a task |
| $WP_c$ | punishment of workers cheating |
| $asp$ | worker's expect payoff |
| $\text{payoff}_i$ | payoff of worker $i$ |
| $pA$ | master's specified probability of audition |
| $a_m$ | the rate of master's reinforcement learning |
| $a_w$ | workers' rate of reinforcement learning |

Auditing mechanism refers to that the master node validates each worker node's returned result. Master node chooses a correct answer and rewards those worker nodes who return correct results by $WB_y$. However, validating process is costly. Thus, master node audits with a probability $p_A(t)$ at time $t$. However, to keep the system always robust, the master node remains a minimum auditing rate $p_A^{\min} > 0$.

Each time the master node audits, it recognizes worker nodes who returns right answer. Thus, to encourage computing, the master node gives each defective worker who returns false results a punishment $WP_c$ in auditing rounds. Thus, payoff of a worker node $i$, $\text{payoff}_i$, is the total reward minus cost (or punishment) in auditing and non-auditing rounds as shown in Eq. 1.

$$\text{payoff}_i = \begin{cases} -WC_t, & \text{if honest, not selected and not audit} \\ WB_y - WC_t, & \text{if honest, selected and not audit} \\ WB_y, & \text{if cheating, selected and not audit} \\ 0, & \text{if cheating, not selected and not audit} \\ WB_y - WC_t, & \text{if honest, audit} \\ -WP_c, & \text{if cheating and audit} \end{cases} \tag{1}$$

### 2.3 Reputation Mechanism

**Reputation Usage.** Reputation is measured by the master node, and the main usage of reputation in our proposed model is to select a correct answer. After collecting all received answers, the master node uses a "voting" method to select a right answer when no auditing mechanism is used.
In this voting mechanism, the master node chooses a correct answer with highest average reputation. For instance, suppose that there are 4 worker nodes with reputation 0.6 return answer "1" and 5 worker nodes with reputation 0.5 return answer "2", then the master node assumes "1" is the correct answer as the average reputation of those 4 worker nodes is higher. When there is a tie, the master node selects the correct answer randomly.

**Comparison of Existing Reputation Algorithms.** In this paper, we consider three different existing reputation algorithms. In the following sections, reputation of worker $i$ at time $t$ is denoted by $repi(t)$.

**Type 1** algorithm is a widely used simple reputation algorithm [13]. As shown in Eq. 2

$$rep_i(t) = \frac{v_i(t) + 1}{aud(t) + 2} \tag{2}$$

**Type 2** algorithm is propose by Christoforou [12]. As shown in Eq. 3

$$rep_i(t) = \epsilon^{aud(t) - v_i(t)}, \epsilon \in (0, 1) \tag{3}$$

**Type 3** is a reputation algorithm inspire by BONIC [14]. As shown in Algorithm 1

**Algorithm 1.** Type 3 Algorithm

**Require:** Transaction of each node $i$;
**Ensure:** Reputation of each node $i$

**Step 1:**
$\beta_i(t) = 0.1$
**if** worker returns correct answer **then**
    $\beta_i(t) = \beta_i(t) * 0.95$
**else**
    $\beta_i(t) = \beta_i(t) + 0.1$
**end if**
**Step 2:**
**if** $\beta_i(t) > A$ **then**
    $rep_i(t) = 0.001$
**else**
    $rep_i(t) = 1 - \sqrt{\frac{\beta_i(t)}{A}}$
**end if**

**Classification-Based Reputation Mechanism.** In order to avoid aforementioned weakness in Type 1–3 reputation mechanisms, we propose a classification- based reputation mechanism, which is referred as **Type 4** in following parts.

In our proposed reputation mechanism, worker nodes are classified into two categories: WorkerA and WorkerB. WorkerA nodes are those node that can be fully trusted (for each WorkerA $i$, $rep_i = 1$) and WorkerB nodes are those with reputation $0 < rep_i < 1$.

**Algorithm 2.** Type 4 Algorithm

**Require:** Returning result of each node, $S_i$;
Number of each worker's accumulated returning correct answer, $CT_i$.
**Ensure:** Reputation and category of each node $i$

```
1: // Step 1: Find WorkerA
2: for all worker node i do
3:    Mark all of the WorkerA nodes
4: end for
5: //Step 2: Choose correct answer
6: if There exists WorkerA nodes then
7:    if all workerA nodes have the same answer then
8:       Choose WorkerA's answer as a correct answer.
9:    else
10:      Audit to find correct answer.
11:   end if
12: else
13:   Vote for a correct answer
14:   Audit to find correct answer with probability p_A
15:   GOTO Step 3
16: end if
17: //Step 3: Reputation Update
18: for all worker node i do
19:   if worker node i returns correct answer then
20:      if worker node i is of WorkerB then
21:         if CT_i >= M then
22:            Node_i promoted as WorkerA node
23:         end if
24:      end if
25:   else
26:      Set Node_i's reputation to 0
27:   end if
28: end for
```

As shown in Algorithm 2, classification-based reputation mechanism consists of three steps: (1) find WorkerA nodes (Lines 2–4); (2) choose correct answer (Lines 6–16); (3) reputation update (Lines 18–28).

### 2.4 Reinforcement Learning

As mentioned above, both master and worker nodes are rational. Thus to maximize correct rate for master node and payoff for worker nodes, reinforcement learning mechanism is introduced.

For master node in MW-model, it adjusts auditing rate according to all worker nodes' reputation. As shown in Eqs. 4 and 5. We use $k$ to find whether the system is safe or not. If the system is considered to be safe, the auditing rate decreases. To ensure the system robustness, the master node maintains a minimum auditing rate $p_A^{min}$. Learning rate $\alpha_m$ indicates the speed of adjusting auditing rate: if $\alpha_m$ is large, then master node adjusts its auditing rate dramatically; if $\alpha_m = 0$, master node never adjusts auditing rate.

$$p_A(t+1) = \min\{1,\ \max\{p_A(t) - \alpha_m(k-T),\ p_A^{min}\}\} \tag{4}$$

$$k = \frac{\sum_{i \in W_c} rep_i(t)}{\sum_{j \in W} rep_j(t)} \tag{5}$$

For a worker node, it wants to maximize its payoff. This payoff depends on its own calculating computing task and master node's auditing actions (see Table 1). After receiving its payoff, each worker $i$ adjusts its cheating rate $pC_i$ by using Eq. 6. If its payoff is higher than its aspiration $a_i$ , then it more prefers this action in the following rounds. $S_i$ indicates the action of worker node $i$ in this round, where $S_i = 1$ means $i$ sends correct result in this round and $S_i = -1$ means it cheats. We assume each worker has the same learning rate $\alpha_w$.

$$pC_i(t) = \max\{0,\ \min\{1, pC_i(t-1) - \alpha_w(payof\, f_i - a_i)S_i\}\} \tag{6}$$

## 3 Simulation and Analysis

In this paper, we perform a simulation in a master-worker network with one master node and nine worker nodes. In each round of task, the worker node first distributes a task, then each worker node $i$ returns a correct answer with the probability $1 - pC_i$ and a false answer with the probability $pC_i$. After worker nodes returning results, the master node performs auditing mechanism with the probability $p_A$. If master node audits, it updates each worker's reputation by different reputation mechanism (see Sect. 2, Type 1–4) and its own auditing rate using Eq. 4. After selecting correct answer, all worker nodes update payoff and cheating rate.

To validate the effectiveness of our proposed mechanism, we perform simulations in the following aspects: (1) fairness of evaluating reputation; (2) robustness under stochastic attacks; (3) cost of defending attacks. Without special emphasis, we use the parameters in Table 2 for following simulations. We mainly compare our results with

**Table 2.** Simulation parameters

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| $WB_y$ | 1 | $WC_t$ | 0.1 |
| $WP_c$ | 0 | $a_i$ in Eq. 6 | 0.1 |
| $p_A(0)$ | 0.5 | $p_A^{min}$ | 0.01 |
| $p_C(0)$ | 0.5 | $a_m$ | 0.1 |
| $a_w$ | 1 | $\epsilon$ (in Type 2) | 0.01 |
| $\tau$ (in Eq. 4) | 0.5 | $M$(in Type 4) | 30 |

three types of reputation mechanisms mentioned in Sect. 2.3, and "Type 4" refers to our proposed reputation mechanism.

### 3.1 Effectiveness on Calculating Reputation

First of all, we capture the reputation dynamics of four different reputation algorithms. Figure 1 presents reputation dynamics of nine rational worker nodes.

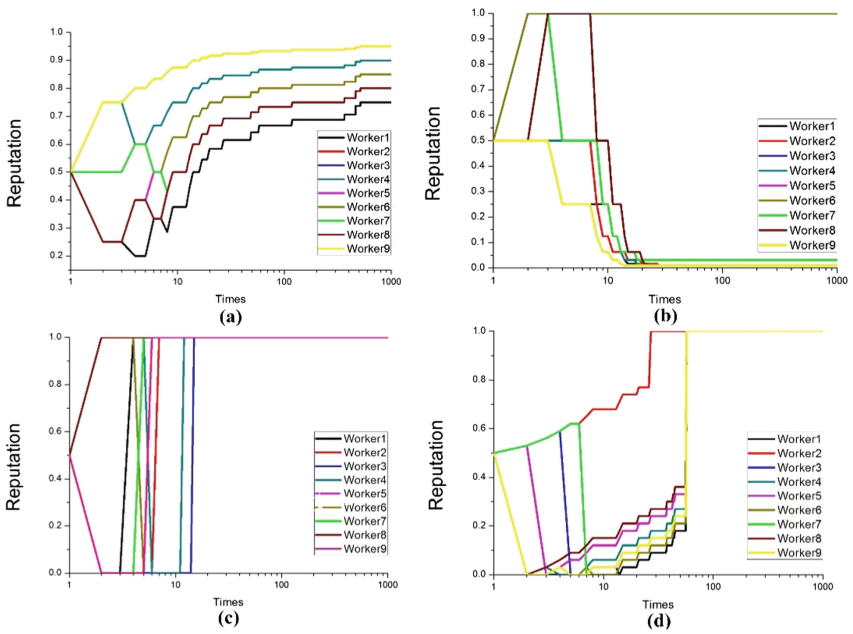


**Fig. 1.** Dynamics of rational worker nodes' repuation. (a) Type 1 (b) Type 2 (c) Type 3 (d) Type 4

In Type 1, 3 and 4 algorithms, finally, all rational worker nodes get a high reputation as they constantly return correct results. Type 1 and 4 algorithm has a stable reputation, however, type 3 algorithm has a dramatically dynamics due to its restrict punishment. In type 2 algorithm, due to the feature of function Eq. 3, once a worker node cheats, it can never raise its reputation again. In our proposed algorithm, there exists a mutation that a rational worker node may

raise its reputation to 1 directly. That is due to its great performance by sending correct results constantly over 30 times. Thus, our proposed reputation mechanism can ensure that rational worker nodes who never cheat can have a very high reputation, so that the fairness of reputation algorithm can be guaranteed.

From the master node perspective, we capture the dynamics of audit times. As shown in Fig. 2, the speed of auditing times increasing decrease along with times. Due to reinforcement learning algorithm, the master node audits less in future rounds. Type 1, 3 and 4 algorithms perform better than type 2 as the total audit times of type 2 are much more than the other 3 algorithms.

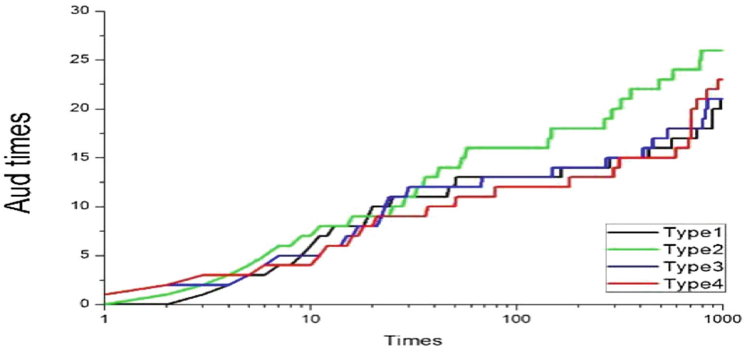


**Fig. 2.** Dynamics of master node's audit times.

Generally, our proposed type 4 algorithm has similar results to other three algorithms on guarantee reputation calculation fairness.

### 3.2 Effectiveness on Defending Attacks

In this section, we mainly discuss the robustness of reputation algorithms under stochastic attack.

A randomly chosen worker node performs as a malicious node, randomly selects 10 computing rounds to perform cheating action in the selected round and 9 rounds following this, total 100 rounds of cheating actions are performed by this chosen worker node.

As shown in Fig. 3, the three existing reputation algorithm have similar results shown in Fig. 1, which means these three algorithms are not sensitive to stochastic attacks. Thus, the malicious worker node that performs this stochastic attack cannot be found.

However, as shown in Fig. 4, our proposed algorithm successfully finds out that worker node 7 is a malicious node. In node 7's cheating rounds, this node's reputation is set to 0. So that a malicious worker is separated. Thus, our proposed model is very sensitive to stochastic attack and more effective on defending stochastic attack compared to other three algorithms.

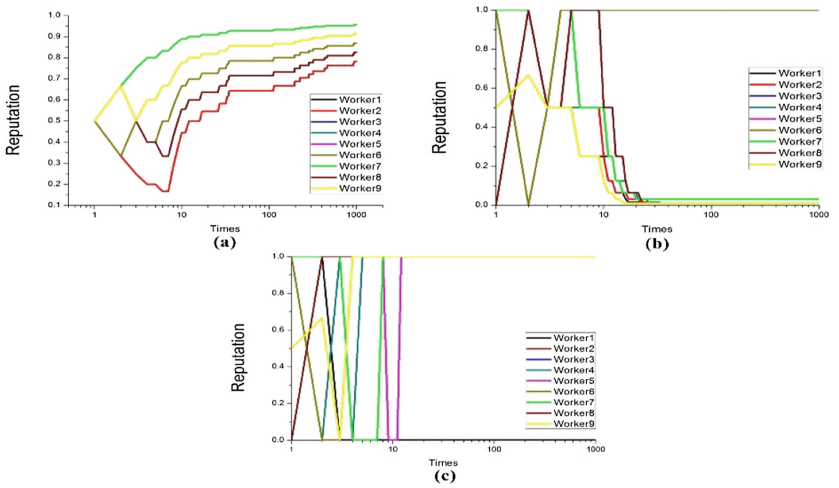


**Fig. 3.** Dynamics of reputation under stochastic attack. (a) Type 1 (b) Type 2 (c) Type 3

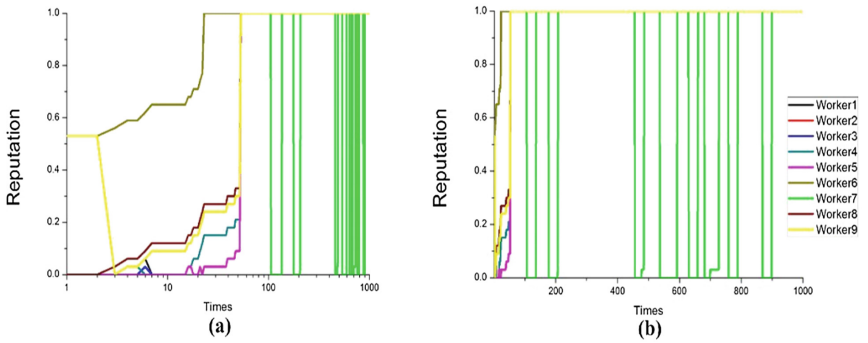


**Fig. 4.** Dynamics of reputation under stochastic attack with type 4 algorithm. (a) Normal version (b) Zoomin version

## 4 Conclusion and Future Work

In this paper, we propose a classification-based reputation mechanism. In our proposed mechanism, worker nodes are classified into two categories, which helps reduce master node's auditing cost. Simulation results show that our mechanism can more effectively induce rational worker nodes return correct answers and malicious nodes can be separated quickly.

Nevertheless, malicious attacks are complex in real systems. For further research, we will consider more realistic scenarios that worker nodes can collaborate to forge results and gain advantages from the master node. Thus, the prevention of more complex attack model is future research direction.

## References


1. Korpela, E.J., et al.: SETI@home-massively distributed computing for SETI. Comput. Sci. Eng. **3**(1), 78–83 (2001)
2. Amazonas Mechanical Turk. https://www.mturk.com
3. Goux, J.P., et al.: An enabling framework for master-worker applications on the computational grid, vol. 4(1), pp. 43–50 (2000)
4. Anta, A.F., Georgiou, C., Mosteiro, M.A., Pareja, D.: Multi-round master-worker computing: a repeated game approach. In: 2016 IEEE 35th Symposium on Reliable Distributed Systems (SRDS), pp. 31–40. IEEE, September 2016
5. Christoforou, E., Anta, A.F., Georgiou, C., Mosteiro, M.A.: Algorithmic mechanisms for reliable master-worker internet-based computing. IEEE Trans. Comput. **63**(1), 179–195 (2014)
6. Nguyen, T.T.H., Brun, O., Prabhu, B.J.: Performance of a fixed reward incentive scheme for two-hop DTNs with competing relays: short talk. ACM SIGMETRICS Perform. Eval. Rev. **44**(3), 39 (2017)
7. Seregina, T., Brun, O., El-Azouzi, R., Prabhu, B.J.: On the design of a reward-based incentive mechanism for delay tolerant networks. IEEE Trans. Mob. Comput. **16**(2), 453–465 (2017)
8. Lu, K., Wang, S., Xie, L., Wang, Z., Li, M.: A dynamic reward-based incentive mechanism: reducing the cost of P2P systems. Knowl. Based Syst. **112**, 105–113 (2016)
9. Gupta, R., Somani, A.K.: Game theory as a tool to strategize as well as predict nodes' behavior in peer-to-peer networks. In: International Conference on Parallel and Distributed Systems (2005)
10. Orset, J.M., Ana, C.: Security in ad hoc networks. In: Ad Hoc Networking Towards Seamless Communications. Springer Netherlands, pp. 756–775 (2002)
11. Ciccarelli, G., Cigno, R.L.: Collusion in peer-to-peer systems. Comput. Netw. **55**(15), 3517–3532 (2011)
12. Christoforou, E., Anta, A.F., Georgiou, C., Mosteiro, Miguel A., Sánchez, A.: Reputation-based mechanisms for evolutionary master-worker computing. In: Baldoni, R., Nisse, N., van Steen, M. (eds.) OPODIS 2013. LNCS, vol. 8304, pp. 98–113. Springer, Cham (2013). https://doi.org/10.1007/978-3-319-03850-6_8
13. Sonnek, J., Chandra, A., Weissman, J.: Adaptive reputation- based scheduling on unreliable distributed infrastructures. IEEE Trans. Parallel Distrib. Syst. **18**(11), 1551–1564 (2007)
14. BONIC reputation platform. http://bonic.berkeley.edu/trac/wiki/
15. Kondo, D., et al.: Characterizing result errors in internet desktop grids. In: European Conference on Parallel Processing, pp. 361–371 (2007)