Multi-round Master-Worker Computing: A Repeated Game Approach

Antonio Fernandez Anta — Institute IMDEA Networks, Spain

Chryssis Georgiou — **University of Cyprus, Cyprus**

Miguel A. Mosteiro — Pace University, NY, USA

Daniel Pareja — Kean University, NJ, USA

SRDS 2016

September 27th, 2016

Budapest, Hungary

Computational Tasks

- Increasing demand for processing complex computational tasks
  - One-processor machines have limited computational resources
  - Powerful parallel machines (supercomputers) are expensive and are not globally available
- Internet emerges as a viable platform for supercomputing
  - Grid and Cloud computing
    - e.g., EGEE Grid, TERA Grid, Amazon's EC2
  - Volunteer Master-Worker computing: @home projects
    - e.g., SETI@home, AIDS@home, Folding@home, PrimeNet
  - Crowd computing
    - e.g., Amazon's Mechanical Turk (human-based computing)

SETI@home by the numbers

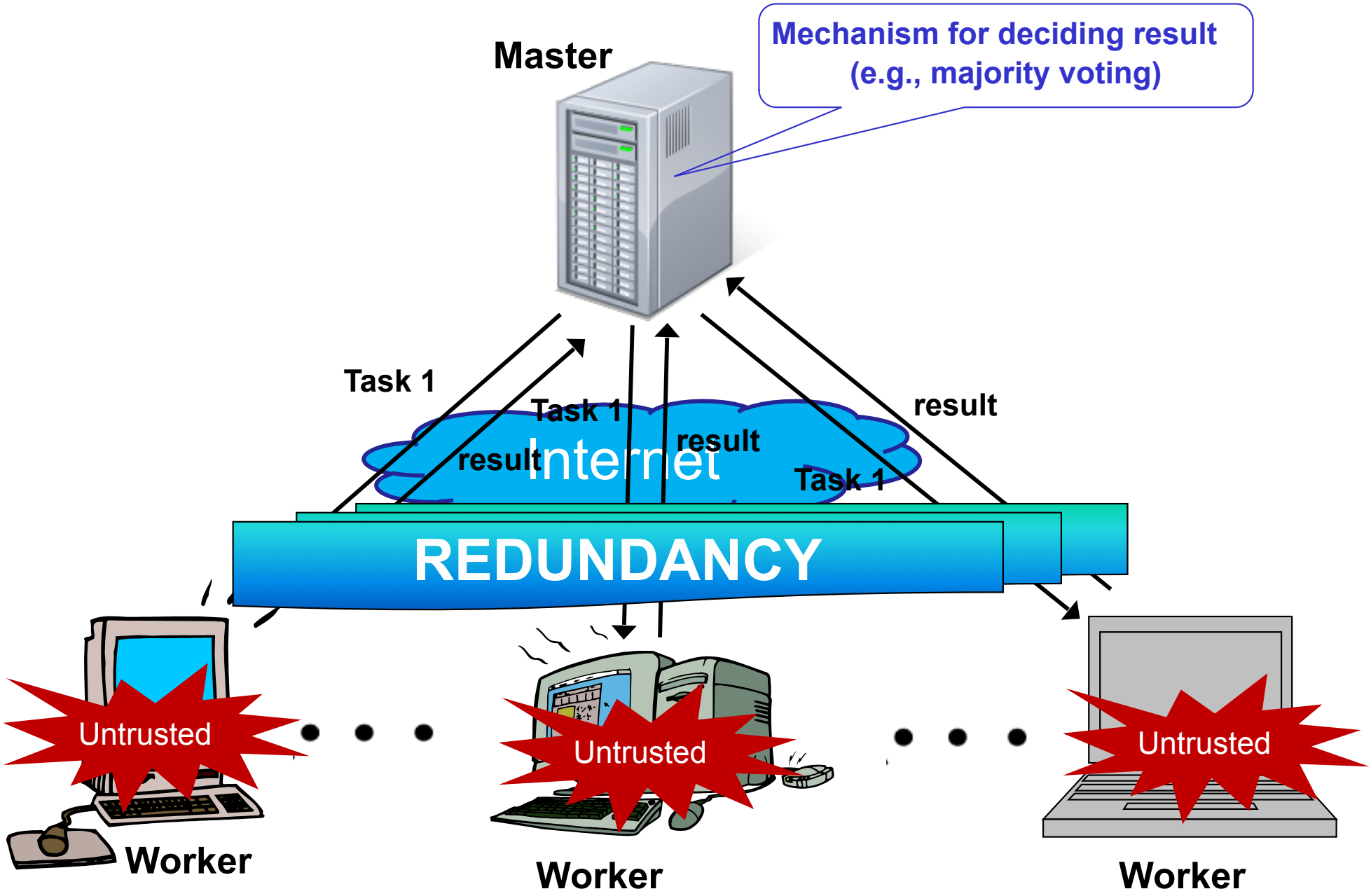
- As retrieved in September 2016
  - 138,473 active CPUs (out of a total of 1.6 million) in 234 countries
  - 747.7 TFLOPs

Comparable processing power with top Supercomputers @ a fraction of the cost!

Globally available!

Great potential limited by untrustworthy entities

SETI-like Master-Worker Computing



Type of Workers

- Classical distributed computing: pre-defined behavior
  - Malicious workers: always return a fabricated incorrect result
  - Altruistic workers: always compute and return a correct result

  Voting protocols are designed (majority rules)

  [Sarmenta 02, FGLS 06/12, KRS 06/15, DKS 13, DKRS 13/15]

- Game-theoretic: workers act upon their best interest
  - Rational workers:i.e., they act selfishly aiming to maximize their own benefit [Shneidman Parkes 03]

  Incentives are used to induce a desired behavior: be honest

  [YLR 05, FGM 08, CFGMS 13, FGMP 15]

- A combination of the two approaches to cope with the co-existence of all three type of workers

  [FGM 10, CFGMS 13, CFGM 14, CFGM 16]

General Task Computing Scheme

- Master assigns a task to n workers
- Workers:
  - Decide whether to cheat or not
  - Malicious/Altruistic: always return incorrect/correct result
  - Rational: Decide to cheat with probability p_C
- Master collects responses and verifies answers with probability p_V
  - If master verifies: Rewards honest workers & penalizes cheaters
  - Otherwise:
    - ➢Accepts response returned by majority
    - ➢Rewards those in the majority

Rewards/Punishments

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

Extracted by Empirical results on SETI-like applications
(BOINC, emBOINC, einstein@home,...)

One-shot Mechanism

- Model the decisions (cheat or not / verify or not) as a game between the master and the workers
- Technical approach: Compute the conditions on the parameters for Nash Equilibria
  - Compute p_V s.t. master obtains correct result at low cost
  - Workers' benefit is maximized for $p_C = 0$
- Tradeoff: Probability of correct result vs cost
- Dealing with many tasks
  - Repeat the one-round mechanism for each different task
  - A decent solution but
  - Does not take advantage of knowledge gained in previous rounds

[Christoforou Fernandez **G** Mosteiro 08, Fernandez **G** Mosteiro Pareja 15]

Multi-round Mechanism

- Evolutionary Dynamics: p_V and p_C are updated after each round
- Technical approach: Use Reinforcement learning to update function of worker profit aspiration and master's tolerance to loss.
  - Eventual correctness: After some rounds, the master obtains the correct task in every round, with minimal verification, while keeping the workers satisfied
  - Workers eventually have $p_C = 0$ in every round
- Tradeoff: Time to correctness vs cost

[Christoforou Fernandez **G** Mosteiro Sanchez 13]

Can a different multi-round approach be used for a more effective repeated interaction between the master and the (rational) workers?

Our Approach: Repeated Games

- We model the repeated interactions between the master and the rational workers as a repeated game
  - It captures the effect of long-term interaction
  - The master obtains the correct task results (whp) from the very first round

- In a round, if workers detect that one worker (or more) has deviated from the agreed strategic choice
  - Change their strategy into the one that maximizes the negative effect they have on the utility of the deviated worker
  - Might negatively affect their own utility, but in long-running interactions this punishment threat prevents workers from deviating

[Osborne Rubinstein 94]

Contributions

- First work to apply the repeated games framework to the master-worker paradigm.
- Devised and analysed two mechanisms
  - First mechanism: Workers' decision is deterministic
    - ➢ To detect deviations, the master only provides the number of different answers at the end of each round
  - Second mechanism: Workers' decision is probabilistic
    - ➢To detect deviations the master provides how many of each answer has received at the end of each round

  We prove the conditions and the cost for the master to obtain the correct task result in every round (whp)
- Experimental evaluation via simulation
  - Superiority over previous approaches

Proportional Punishment

- To implement peer-punishment we use *proportional punishment*
  - The penalty for cheaters is proportional to the number of cheaters

- Let F be the set of workers caught cheating in a round that the master verifies. Then

  penalty for each worker in F *is* $WP_C \cdot |F|$

Deterministic Decision Mechanism - Master

```
1 while true do
2     send a computational task to all the workers in W
3     upon receiving all answers do
4         with probability p_V, verify the answers
5         if the answers were not verified then accept the
          majority
6         reward/penalize accordingly
7         send to the workers the number of different answers
          received
```

Deterministic Decision Mechanism - Workers

```
1  strategy ← C̄
2  while true do
3      upon receiving a task do
4          if strategy= C̄ then
5              compute the task and send the result to the
               master
6          else
7              do not compute and send a bogus result to the
               master
8          upon receiving from the master the number of
           different answers do
9              if number of different answers > 1 then
10                 strategy ← C
```

Main Result

For

$$WC_T/(WB_A + WP_C(n/2) < p_V < (WB_A + WC_T)/(2WB_A + nWP_C)$$

$$WB_A > WC_T \text{ and } WP_C > WC_T$$

the mechanism guarantees that the master obtains the correct task result in every round

Utilities for worker i and master in every round

$$U_i = WB_A - WC_T$$

$$U_M = MB_R - nMC_A - p_V MC_V$$

Probabilistic Decision Mechanism

- Much more involved
  - Actually probability used by each worker cannot be inferred accurately from one round
  - Instead, it is possible to provide stochastic guarantees, either from many computations of one worker, or one computation by many workers.

- Need the master to announce how many of each answer has received for workers to detect deviations based on the probability of such outcome

Probabilistic Decision Mechanism - Master

```
1 while true do
2 |   send a computational task to all the workers in W
3 |   upon receiving all answers do
4 |   |   with probability pv, verify the answers
5 |   |   if the answers were not verified then accept the
  |   |   majority
6 |   |   reward/penalize accordingly
7 |   |   send to the workers a list of pairs ⟨answer, count⟩
```

Mixed Probabilistic Decision - Workers

```
1  maxrounds ← ⌊3npc ln(1/ε)⌋                    // Punishment
   decisions only for δ ≥ 1/npc (cf. Lemma 3).
2  counts ← empty queue of integers  // counts[i] is the
   (i + 1)th item, for i = 0, 1, 2, ....
3  for each round = 1, 2, ... do
4      upon receiving a task do
           // computation phase
5          cheat ← { true,  with probability pc
                   { false, with probability 1 − pc
6          if cheat = false then result ← task result
           computed
7          else result ← bogus result
8          send result to the master
           // punishment phase
9          upon receiving from the master a list of pairs
           ⟨answer, count⟩ do
               // update # of cheaters per round
10             verify all answers
11             #incorrect ← number of incorrect answers
12             enqueue #incorrect to counts
13             if size of counts > maxrounds then dequeue
               from counts
               // punishment decision
14             cheaters_min ← n
15             cheaters_max ← 0
16             R ← min{maxrounds, round}
17             for r = 1 to R do
18                 if counts[R − r] < cheaters_min then
                   cheaters_min ← counts[R − r]
19                 if counts[R − r] > cheaters_max then
                   cheaters_max ← counts[R − r]
20                 δ ← √(3 ln(1/ε)/(rnpc))
21                 if δ < 1 then
22                     if cheaters_min ≥ ⌈(1 + δ)npc⌉ or
                       cheaters_max ≤ ⌊(1 − δ)npc⌋ then
                       // Lemma 3
23                         pc ← 1                  // Lemma 4
```

Main Result

For

$$0 < p_C < 1/(2(1+\xi)), \text{ for some } 0 < \xi \leq 1$$

$$p_V > 2WB_A/(2WB_A + nWP_C), \;\; p_V \geq WC_T/WB_A, \text{ and}$$

$$p_V \geq \frac{e^{-n\xi^2/(6(1+\xi))} - \varphi}{e^{-n\xi^2/6(1+\xi))} - p_C^n}, \text{ for some } \varphi > 0$$

the mechanism guarantees that the master obtains the correct result in every round with probability at least

$$1 - \varphi$$

Simulations

- Compare three mechanisms
  - Ours (mixed equilibria) – RG
  - Repeated One-Shot mechanism – ROS
  - Evolutionary Dynamics mechanism – ED

- Choice of parameters values
  - Used in prior works which are consistent with statistics obtained in BOINC projects (SETI@home)
  - Satisfy constraints obtained by their theoretical analyses
  - Make the comparison of the three mechanisms fair

- Run for 200 rounds with up to $n/2$ deviators

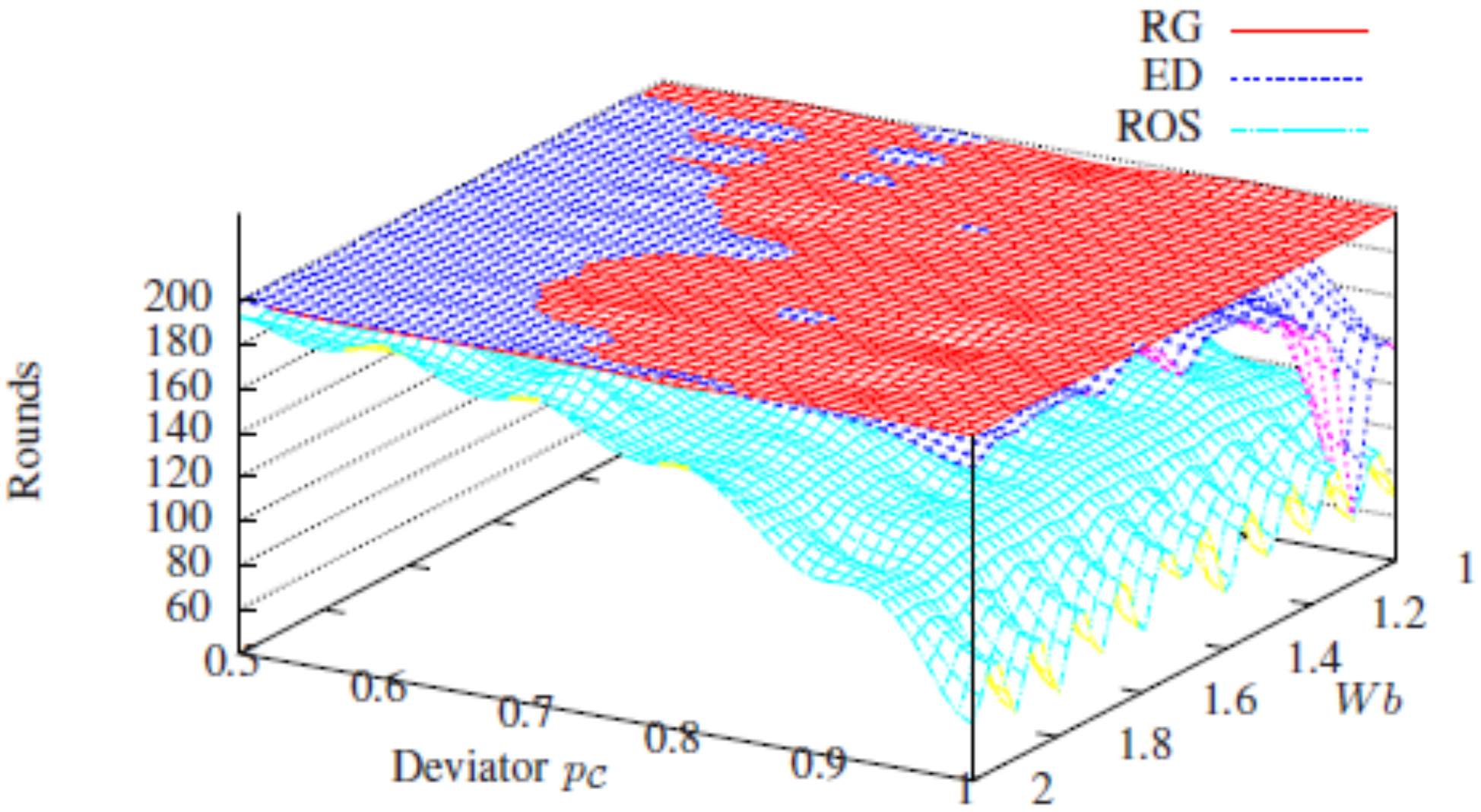
Simulation Parameters

	RG [this paper]	ROS [20]	ED [13]
n	$\{9, 27, 81\}$	$\{9, 27, 81\}$	$\{9, 27, 81\}$
$WB_{\mathcal{A}}$	$\{1, 1.1, \ldots, 2\}$	$\{1, 1.1, \ldots, 2\}$	$\{1, 1.1, \ldots, 2\}$
$WP_{\mathcal{C}}$	$\frac{WB_{\mathcal{A}}}{np_{\mathcal{C}}}$ if $\lvert F\rvert < n$, 0 if $\lvert F\rvert = n$	$WB_{\mathcal{A}}$	$WB_{\mathcal{A}}$
$WC_{\mathcal{T}}$	0.1	0.1	0.1
$p_{\mathcal{C}}$	$\lfloor n/2\rfloor$:0.1, $\lceil n/2\rceil$: $\{0.5, 0.6, \ldots, 1\}$	$\lfloor n/2\rfloor$:0, $\lceil n/2\rceil$: $\{0.5, 0.6, \ldots, 1\}$	$\lfloor n/2\rfloor$:0, $\lceil n/2\rceil$: $\{0.5, 0.6, \ldots, 1\}$
$p_{\mathcal{V}}$	0.17	$\frac{WB_{\mathcal{A}}+0.1}{3WB_{\mathcal{A}}} + 0.01$	initially: 0.5, min: 0.01
$MC_{\mathcal{A}}$	$WB_{\mathcal{A}}$	$WB_{\mathcal{A}}$	$WB_{\mathcal{A}}$
$MC_{\mathcal{V}}$	$nWB_{\mathcal{A}}$	$nWB_{\mathcal{A}}$	$nWB_{\mathcal{A}}$
$MP_{\mathcal{W}}$	$nWB_{\mathcal{A}}$	$nWB_{\mathcal{A}}$	$nWB_{\mathcal{A}}$
$MB_{\mathcal{R}}$	$nWB_{\mathcal{A}}$	$nWB_{\mathcal{A}}$	$nWB_{\mathcal{A}}$
other	$\varepsilon = 0.01$	–	$\tau = 0.5$, $a_w = 0.1$, $\alpha_m =$ $\alpha_w = 0.01$

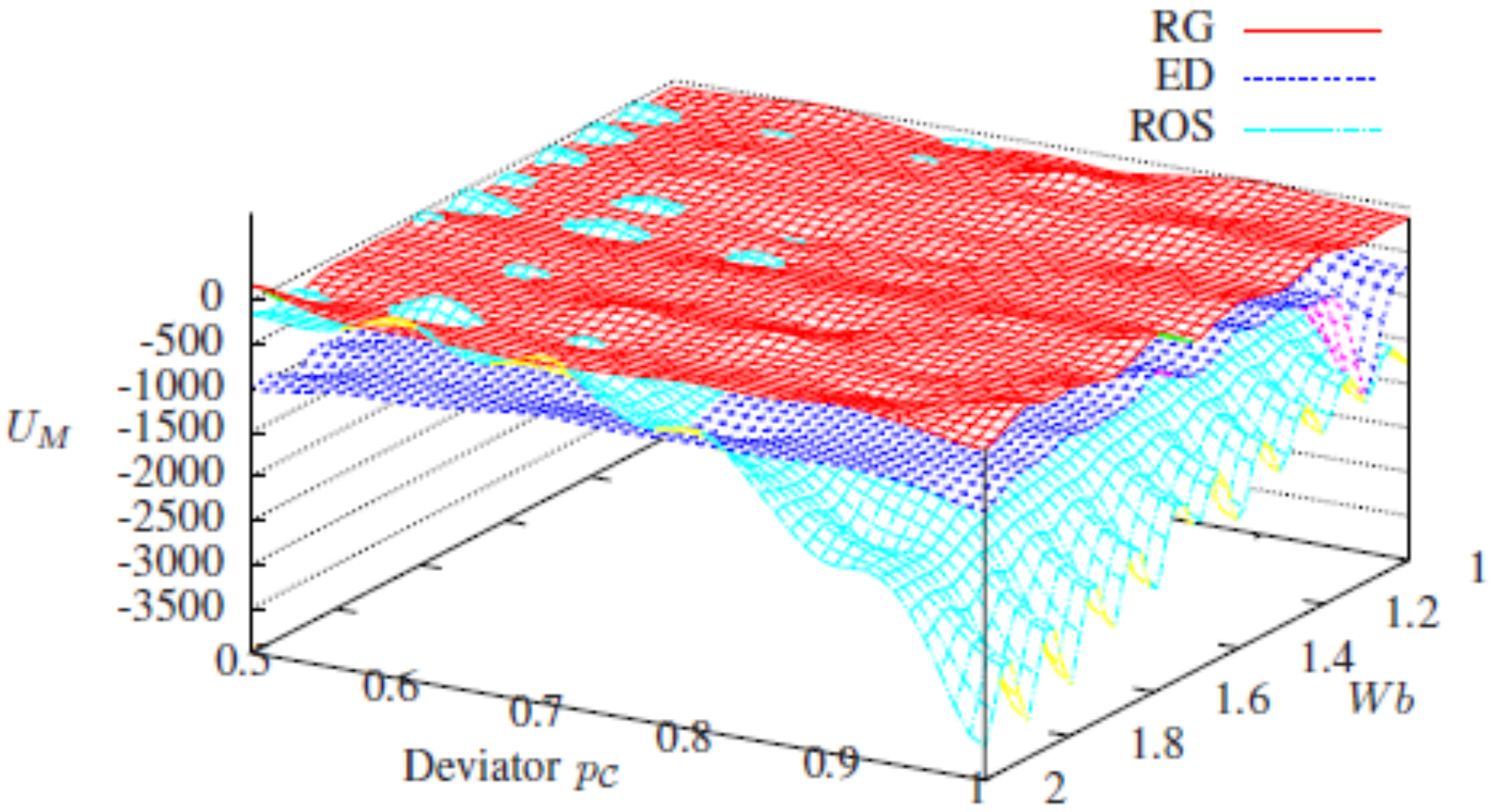
Overall Outcome

- In the presence of up to $n/2$ deviators, our mechanism
  - Performs similarly or better than the evolutionary dynamics approach
  - Both mechanisms perform significantly better than the repeated one-shot mechanism

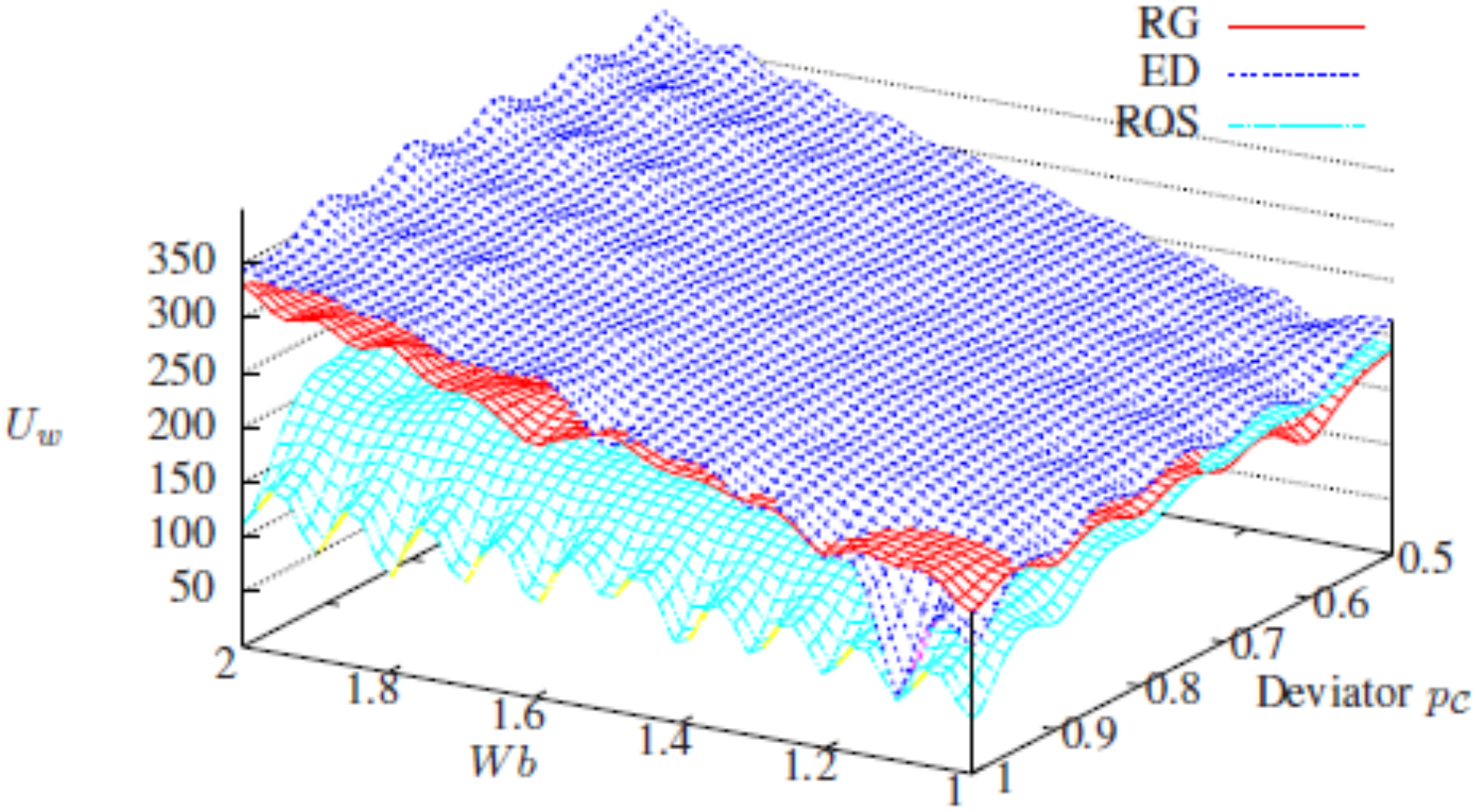
Number of “Correct” Rounds



Cumulative Master Utility



Cumulative Worker Utility



Ongoing & Future Work

- Extend the mechanism to also cope with malicious workers
  - Use statistical information on the different worker types

    [Christoforou Fernandez **G** Mosteiro 14]
  - Challenge: Analysis significantly be revised

- Consider worker collusions [Fernandez **G** Mosteiro Pareja 15]
  - Challenge: The master must cope with collusions without knowing which specific workers are colluding.

- Consider pool of workers to choose from
  - Tolerate workers not responding (e.g., abstaining)
  - Use reputation schemes [Christoforou Fernandez **G** Mosteiro 16]

KÖSZÖNÖM!

Contributions (1)

- First work to apply the repeated games framework to the master-worker paradigm.

- Demonstrate the benefit and promise of this approach
- Under certain conditions
  - Master obtains correct results (whp) from the first round
  - It does so with lower cost than previous approaches

Contributions (2)

- Devised two mechanisms
  - First mechanism (pure equilibria strategies)
    - ➢ workers' decision is deterministic
    - ➢ To detect deviations, the master only provides the number of different answers at the end of each round
  - Second mechanism (mixed equilibria strategies)
    - ➢ workers' decision is probabilistic
    - ➢ To detect deviations more information is needed
      - Which answers the master received
      - And how many of each

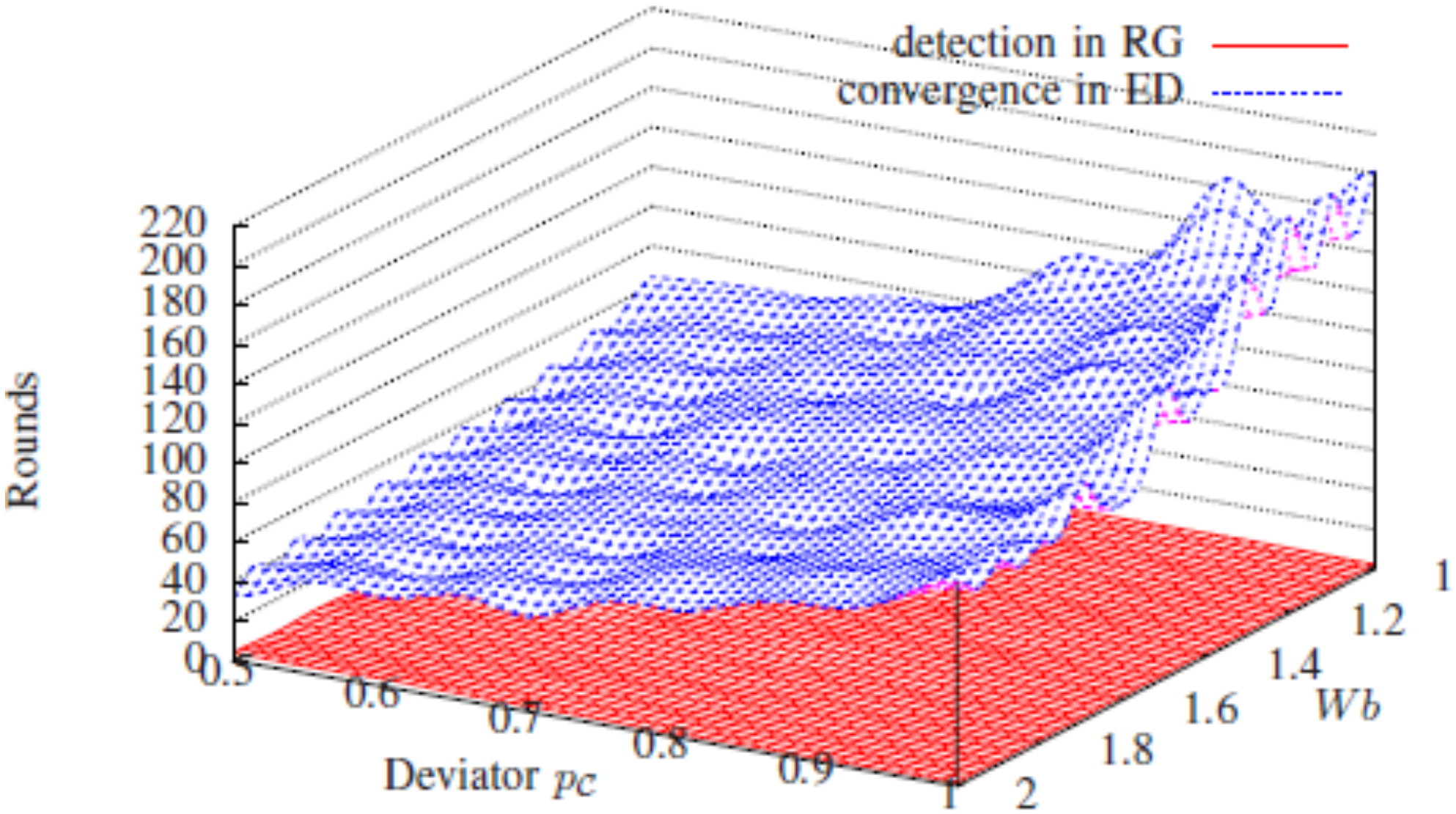
We prove the conditions and the cost under which the Master obtains the correct task result in every round (whp)

Contributions (3)

- Experimental evaluation via simulation
  - Provides insights on the effectiveness of the mechanisms on various parameter values
  - Comparison with the previous approaches (repeated one-shot and evolutionary dynamics)

- In the presence of up to $n/2$ deviators, our mechanism
  - Performs similarly or better than the evolutionary dynamics approach
  - Both mechanisms perform significantly better than the repeated one-shot mechanism

Rounds to Deviation Detection (RG) /Convergence (ED)



Top Three Supercomputers (June 2016)

- Sunway TaihuLight, National Supercomputer Center in Wuxi, China
  - 40,960 Sunway SW26010 260-core 1.45 GHz: 10,649, 600 cores
  - 125,436 TFLOPS (125.4 PetaFLOPS)

- Tianhe-2 (MilkyWay-2), National Supercomputer Center in Guangzhou, China
  - 260,000 Intel Xeon E5-2692 12-core 2.200GHz: 3,120,000 cores
  - 54,902 TFLOPS (54.9 PetaFLOPS)

- Titan Cray XK7, Cray Inc, USA
  - 35,040 Opteron 6274 16-core 2.200GHz: 560,640 cores
  - 27,112 TFLOPS (27.1 PetaFLOPS)

€ € € € € € €

https://www.mturk.com/mturk/welcome

amazonmechanical turk beta
Artificial Artificial Intelligence

Your Account | HITs | Qualifications

Already have an account?
Sign in as a Worker | Requester

Introduction | **Dashboard** | **Status** | **Account Settings**

Mechanical Turk is a marketplace for work.

We give businesses and developers access to an on-demand, scalable workforce.
Workers select from thousands of tasks and work whenever it's convenient.

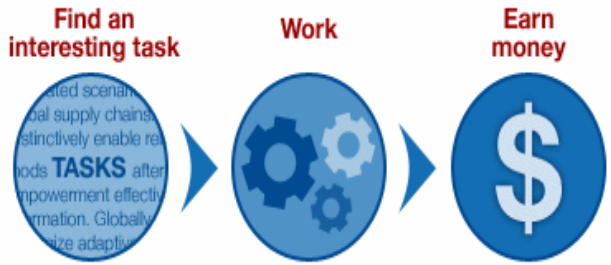
289,486 HITs available. View them now.

Make Money
by working on HITs

HITs - *Human Intelligence Tasks* - are individual tasks that you work on. Find HITs now.

As a Mechanical Turk Worker you:

- Can work from home
- Choose your own work hours
- Get paid for doing good work



Find HITs Now

or learn more about being a **Worker**

Get Results
from Mechanical Turk Workers

Ask workers to complete HITs - *Human Intelligence Tasks* - and get results using Mechanical Turk. Register Now

As a Mechanical Turk Requester you:

- Have access to a global, on-demand, 24 x 7 workforce
- Get thousands of HITs completed in minutes
- Pay only when you're satisfied with the results

Get Started

An amazon.com company