TASK DISPATCH THROUGH ONLINE TRAINING FOR PROFIT MAXIMIZATION AT THE CLOUD

Sowndarya Sundar and Ben Liang

IEEE INFOCOM Workshop on Network Intelligence

29th April, 2019

The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

Computational Offloading

- Improves response time of computation-intensive tasks
- Improves mobile device's battery lifetime

Problem: Where to schedule the tasks at the cloud?

- Heterogeneous cloud servers
- Server load constraints
- Online task arrivals

Objective:

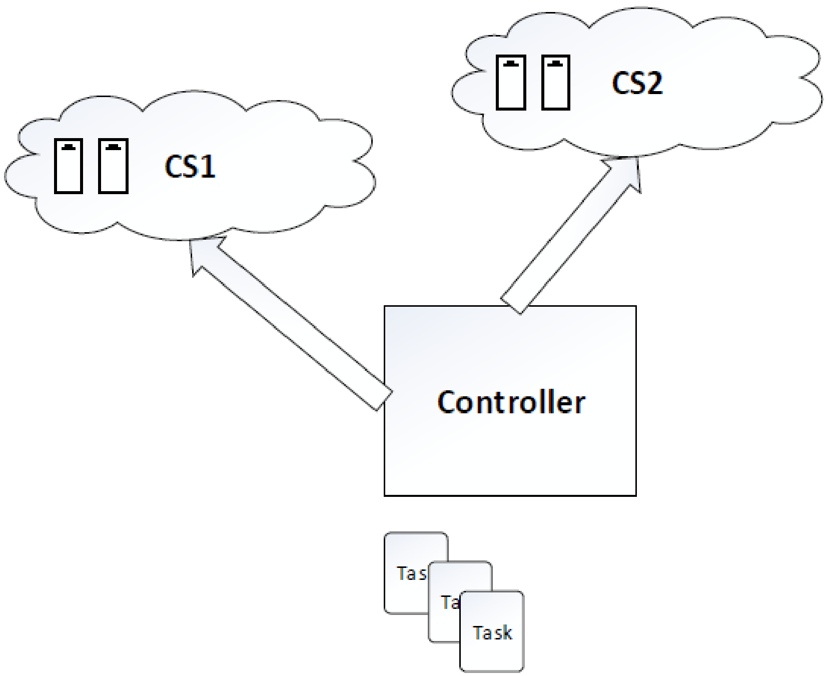
- To maximize profit for a cloud service provider (CSP)

Related Work

- Offline problems [Zhou ,2015], [Mao, 2017], [Chen, 2018]
- Online problems
  - Fluid tasks [Liu, 2016], [Goudarzi, 2011]
  - Homogeneous resources [Peng, 2015], [Champati, 2017]
  - Various objectives: makespan [Fang, 2010], response time [Liu, 2016]
  - Often heuristic solutions

System Model and Problem Formulation

Cloud Servers and Task Arrival



CSP with cloud servers (CSs)

- Processor r in CS k generates profit p_{rk} per unit time.
- Tasks arrive at CSP's controller at average rate λ tasks per unit time.
- Task j requires processing time t_{jrk} on processor r in CS k.
  - Known only once the task arrives at the controller.

Task Scheduling

- Controller distributes tasks among processors
- Decision variables:

  $x_{jrk} = 1$, if task j is to be scheduled on processor r of CS k, and 0 otherwise.

- Task scheduling constraints:

(1) Load (total execution time) constraints L on each processor

(2) Each task can be executed on at most one processor

Online Profit Maximization

Unknown

Unknown

$$\underset{\{x_{jrk}\}}{\text{maximize}} \quad \sum_{j=1}^{M} \sum_{k=1}^{K} \sum_{r=1}^{P_k} p_{rk} t_{jrk} x_{jrk}$$

Total profit

$$\text{subject to} \quad (1) - (2),$$

$$x_{jrk} \in \{0,1\} \quad \forall j \in \{1, \ldots, M\},$$

$$k \in \{1, \ldots, K\}, r \in \{1, \ldots, P_k\}.$$

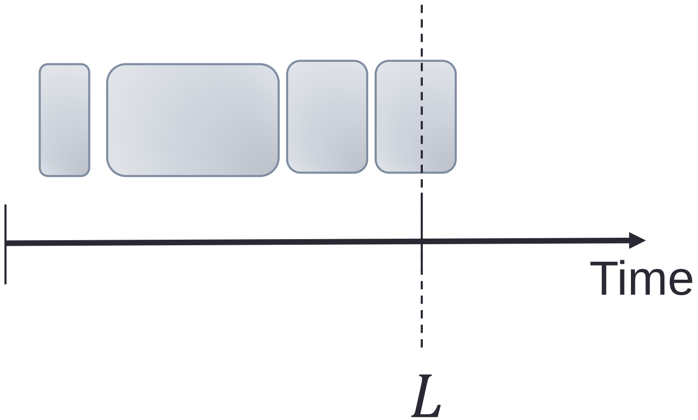
- To maximize total profit over duration T
- Total number of tasks M that arrive within T is random.
- We also do not know the processing times t_{jrk} in advance.

The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

Task Dispatch through Online Training (TDOT)

TDOT Outline

- Consists of a **training** phase and an **exploitation** phase.
- Balanced by a user-defined parameter, $0 < \epsilon < 1$
- Allows profit to be collected from **partially-completed** tasks



Processor load constraint

Training Phase

Expected number of tasks: $E[M] = \lambda T$

Average rate

System duration

- Observe the first $\lfloor \epsilon \lambda T \rfloor$ arriving tasks : **training set A**
- Allocate ϵL load constraint to A
- Arbitrarily schedule these tasks
  - Only to learn the characteristics of tasks

Training Phase (cont.)

- Find **weights** (Lagrange multipliers)

$$\mathbf{u}^* = \mathrm{argmin}_{\mathbf{u} \geq 0} \underbrace{D(\mathbf{u}, \mathcal{A})}_{\boxed{\text{Dual of binary-relaxed offline problem}}}$$

- One weight value for each processor.
- Contains information about the **resource demand with respect to the processor load**.

Exploitation Phase

- Applied to non-training set of tasks A^c
- For each arriving task, we use weights $\boldsymbol{u}$ to obtain the scheduling decision:

$$\underbrace{(r', k')}_{\text{Chosen Processor}} = \operatorname*{argmax}_{r,k} \underbrace{(p_{rk} - u^*_{rk})\, t_{jrk}}_{\text{(Unit profit – Weight) x Processing time}}$$

- Apply remaining load constraint: $(1 - \epsilon)L$

Properties of TDOT

- Single-shot algorithm: no iteration
- Efficient solution: linear programming
- Particularly useful if tasks arriving within duration T have similar characteristics.
- Can prove a performance bound for i.i.d. tasks.

Performance Bound

Theorem 1. *If* $\overbrace{\frac{OPT}{c_{max}} \geq K P_{max} \frac{\ln(K^2 P_{max}^2/\epsilon)}{\epsilon^3}}^{\text{Easily-met condition}}$, *then we have*

$$\underbrace{\mathbb{E}_M\left[\mathbb{E}[S(\boldsymbol{u}^*, \mathcal{A}^c)|M]\right]}_{\text{Expected TDOT profit on non-training set}} \geq \left(1 - 2\epsilon - \epsilon\sqrt{\lambda T}\,\mathbb{E}_M\left[\frac{1}{\sqrt{M}}\right]\right) \underbrace{OPT}_{\text{Offline optimal profit}}.$$

- OPT: maximum profit over T time slots (offline)
- c_{max}: Max. profit per unit time
- K: Number of CSs
- P_{max}: Max. number of processors in a CS

Performance Bound (cont.)

Theorem 1. *If* $\frac{OPT}{c_{max}} \geq KP_{max}\frac{\ln(K^2P_{max}^2/\epsilon)}{\epsilon^3}$, *then we have*

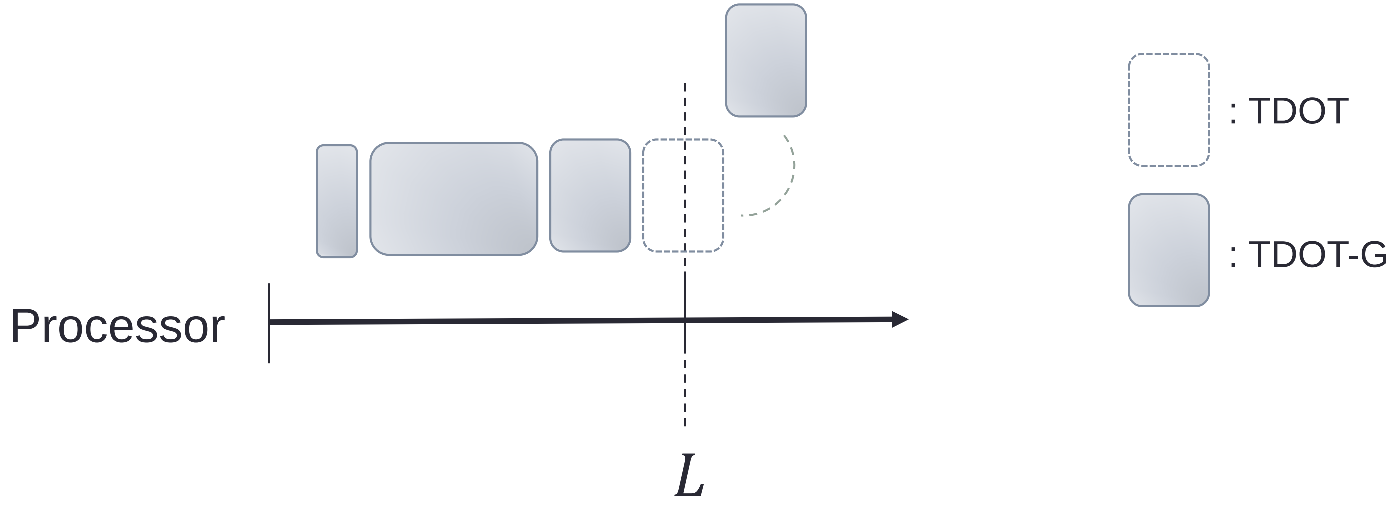
$$\mathbb{E}_M[\mathbb{E}[S(\boldsymbol{u}^*, \mathcal{A}^c)|M]] \geq \left(1 - 2\epsilon - \epsilon\sqrt{\lambda T}\mathbb{E}_M\left[\frac{1}{\sqrt{M}}\right]\right) OPT.$$

Performance depends on $0 < \epsilon < 1$: proportion of tasks in training set, i.e., $\frac{|A|}{E[M]}$.

TDOT with Greedy Scheduling (TDOT-G)

TDOT-G

- Profit only obtained from **fully-completed** tasks
- If task cannot be scheduled on the **best** processor, we try the **second best**, **third best,** and so on.



Time Complexity Analysis

- Training phase : $O((|\mathcal{A}|P)^{3.5})$

  where P is total number of processors.

- Exploitation phase :

  $O((1-\epsilon)M)$ for TDOT.

  $O((1-\epsilon)MP)$ for TDOT-G.

Trace-Driven Simulation Results

Comparison Targets

- **Greedy Algorithm:** greedily schedules each task to processor with maximum profit
- **Logistic Regression (LR):** treats profit maximization as a classification problem
- **Logistic Regression – Greedy (LR-G):** hybrid technique
- **Upper Bound Offline:** upper bound to the optimal solution, assuming all task information is known in advance.

Simulation Setup and Task Times

- We use publicly-available **Google cluster data**
- We use the **task events information** to obtain task arrival times, and compute average arrival rate λ.
- We use **task usage information** to obtain task execution times.
- Assume the number of tasks, M, is Poisson with rate λT.
  - Unknown a priori
  - Task arrival process unimportant

Profit on Non-Training Set

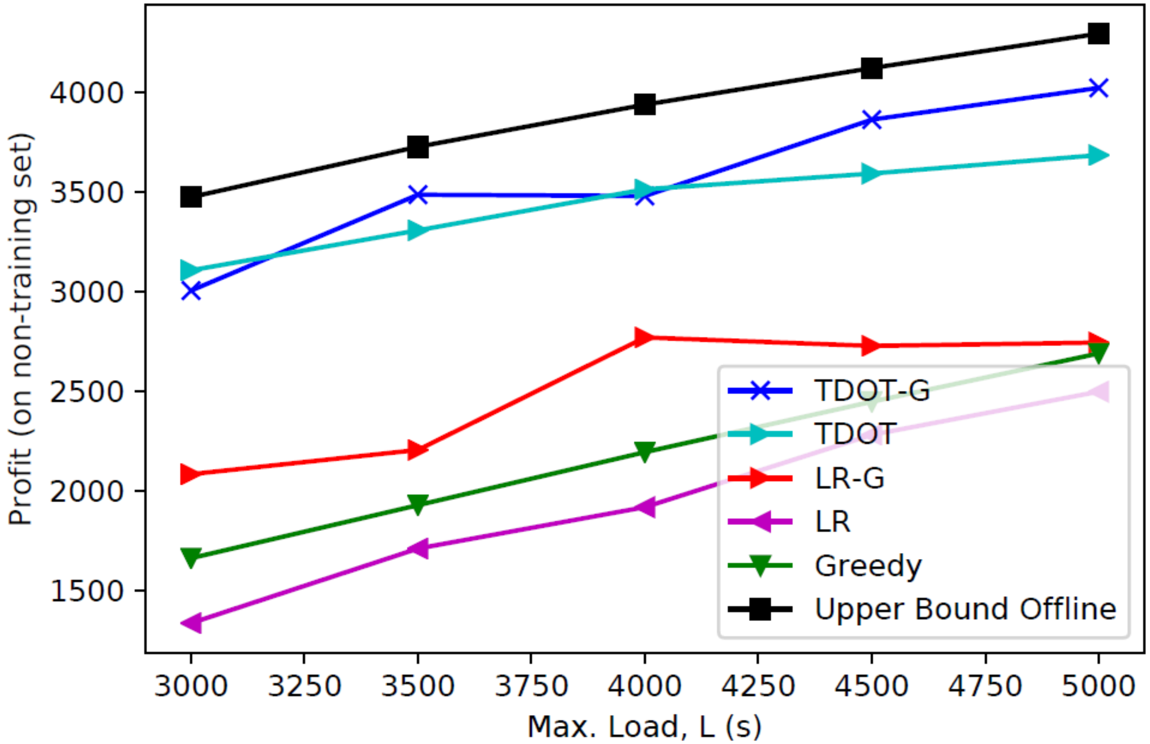


Fig. 1: Effect of max. load L on non-training set profit

Overall Profit

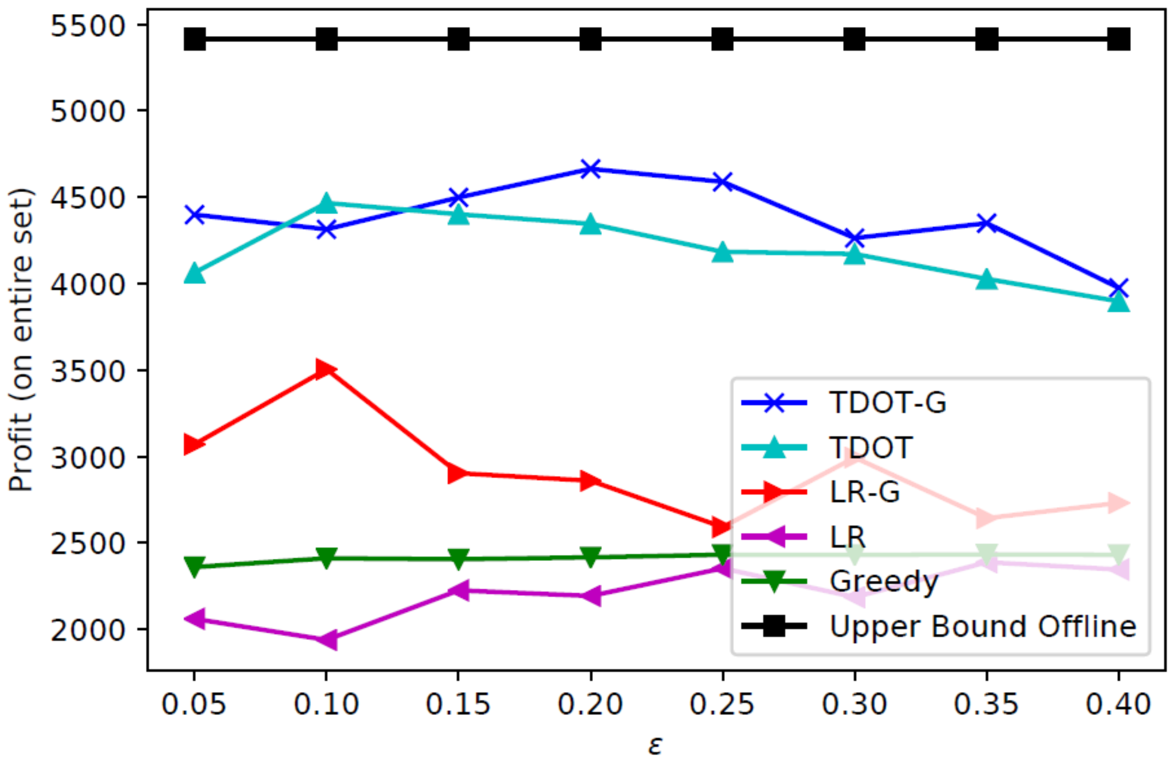


Fig. 2: Effect of ϵ on overall profit

Observations

- TDOT more effectively transfers information from training to non-training set
- Picking the right proportion of training data is important (~20% of expected number of tasks).
- Greedy scheduling of edge cases improves performance (TDOT-G > TDOT)

Summary

- Task scheduling to **maximize profit** under load constraints and **online task arrivals.**
- Proposed TDOT and TDOT-G, consisting of **training** and **exploitation** phases.
- **Low-complexity** solutions
- **Performance bound** for TDOT
- **Trace-driven simulation** demonstrates superior performance