Learning-Based Localized Offloading with Resource-Constrained Data Centers

Jia Guo, James B. Wendt, Miodrag Potkonjak
Computer Science Department
University of California, Los Angeles
Email: {jia, jwendt, miodrag}@cs.ucla.edu

***Abstract*—Offloading has emerged as a new paradigm to save energy for mobile devices in the context of cloud computing systems. Unlike the traditional cloud computing, it offers the flexibility of switching between local and remote execution, and employs accurate profiling of tasks. Given a resource-constrained data center, an interesting optimization question is which tasks should be offloaded/run locally so that global energy savings is maximized. The main technical difficulties are related to the uncertainty and variability of congestion, as well as the need for a real-time, low overhead and localized decision procedure that are near optimal. We introduce a combination of statistical and learning-based techniques that use the results of offline centralized algorithms to create localized online solutions that perform well under realistic workloads. The procedures and algorithms are compared with upper bounds to demonstrate their effectiveness.**

***Keywords*-offloading; scheduling; online localized decision; classification;**

I. Introduction

Energy is one of the biggest constraints for the development of mobile technologies. In recent years, researchers have proposed various frameworks that enable offloading applications to the cloud as a way of saving energy [1] [2] [3]. These frameworks promise to drastically reduce the energy consumption emerging scenarios like gaming, context sensing [4], mobile health [5] and various computer vision related applications. One difference from the traditional cloud computing is that offloading offers the flexibility of switching between local and remote execution. Thanks to the flexibility, data centers serving offloading frameworks do not need to be over-provisioned to ensure the service for every single task. Instead, it can select only the efficient tasks based on the fine grained profiling employed by offloading frameworks. Consider the utilization rate of only 10-50% in today's data centers [6], the reduction in the cost can be huge.

The key in designing such a system involves solving an optimization problem: given data center resource constraints, how can we decide which tasks to offload such that the global energy savings are maximized. The system is desired to be localized, i.e., an optimal offloading decision should be made by mobile devices without communicating with the data center to avoid the unnecessary energy overhead. The system also needs to be light weight to enable occasional update, when major change in the workload happens. Another difficulty involves the uncertainty and variability of congestion.

Table I: Task Model

Parameter	Explanation	Example
τ^l	Local exec. time	300s
τ^c	Cloud exec. time	20s
τ^t	Transmission time	20s
E^l	Local exec. energy	300J
E^c	Offload energy overhead	20J
t^r	Release time	36000s
t^d	Deadline	36400s

Our solution is able to satisfy all the aforementioned requirements by learning online localized decisions from offline centralized solutions. Contributions of this paper are as follows. First, we are the first to propose a learning-based localized solution to the resource allocation problem in offloading systems. Second, we propose a novel time-efficient offline scheduling algorithm that achieves near-optimal results. Third, we use real life work load traces to build realistic simulation of the offloading system.

II. Preliminaries

Consider a scenario where various users offload tasks to the cloud for remote execution. The users neither communicate nor interfere with each other. The data center has a limited number C of machines, and we assume each machine runs a single task in a non-preemptive fashion. The tasks have a release time and a deadline, and are also characterized by a set of parameters derived from established computation, network, and energy models [7]. Table I shows the parameters and exemplary values. We further define *benefit* b of a task to be the energy savings per unit cloud execution time.

Since no offloading systems exist in full production, there is currently limited available real usage data. We can only assume similarity between some aspects of existing systems and the offloading system. We examined the traces of Google computing clusters [8] and adopted power law and log normal distribution for the amounts of computation and data size of tasks respectively. We use real-life cellular network traces for the arrival model of offloading tasks, with the assumption that the variance and unpredictability of the offloading workload can be characterized by the traces.

In offloading, the tasks can either run locally or be executed remotely in the cloud. Define $D_i^l \in \{1, 0\}$ to be the local decision on whether task T_i is offloaded to the cloud, and $D_i^c \in \{1, 0\}$ to be whether the data center executes T_i. The energy savings ΔE_i for the task will be in the following three states depending on the decisions: if successfully offloaded and executed in the data center, $\Delta E_i = E_i^l - E_i^c$; if offloaded yet not scheduled due to resource limits $\Delta E_i = -E_i^c$; if locally executed (without being offloaded) $\Delta E_i = 0$. Our goal, therefore, is to find an online localized decision process for $D_i^l(t)$ and $D_i^c(t)$ at time slot t such that the overall energy savings $\sum \Delta E_i$ is near-maximal. Notice that the time also plays an important role in the decision, and thus we annotate the decision with t as a parameter.

III. Method

To solve the optimization problem, we first examine it in an offline environment. We formulate our problem as a scheduling problem, and show upper bounds of the solution to the problem. We then propose an algorithm that achieves near-optimal results by using statistical techniques. The information from offline scheduling is extracted by classification, which enable local users to make localized real-time online decision for offloading.

A. Offline Scheduling

1) The Scheduling Problem: The offline phase assumes centralized control and the knowledge of all the tasks. Despite the complicated models and procedures, the core task of the offline phase is equivalent to the following: we want to decide when and which tasks to be scheduled locally or in the cloud, such that the energy is maximized. Thus the offline problem can be simplified to a scheduling problem of minimizing the number of weighted tardy jobs [9], where the weight corresponds to the energy savings ΔE_i of the task. The simplification is justified because the offload procedure ($D_i^l(t)$ and $D_i^c(t)$) no longer matters, as long as there's one final decision $D_i(t)$ for each task on whether and when it's executed in the data center. The scheduling problem is NP-hard [9]. It is apparently not feasible to obtain the optimal results even in the offline settings. Therefore, we propose a polynomial near-optimal algorithm to tackle the problem.

2) Baselines and Upper Bound: We use two baseline algorithms: *Deadline* and *Greedy*. Deadline refers to the earliest deadline algorithm, which is proved to be optimal in unweighted scheduling problems. The Greedy baseline orders tasks according to their benefit b and schedules each task at the earliest time slot available.

Based on the upper bound proposed by Bar-Noy et al. [10], we propose a faster linear programming (LP) based upper bound to show the effectiveness of our offline algorithm. The LP relaxes the problem by a) allowing preemptive scheduling; and b) allowing a fraction of a task to be scheduled. The formulation of LP is as follows. Note that x_{it} is 0 for $t \notin [t_i^r, t_i^d)$.

$$\begin{aligned}
\underset{x_{it}}{\text{maximize}} \quad & \sum_{i=1}^{N} \sum_{t=t_i^r}^{t_i^d-1} b_i \cdot x_{it} \\
\text{subject to} \quad & \\
& \sum_{t=t_i^r}^{t_i^d-1} x_{it} \le \tau_i^c, \text{ for each } i \\
& \sum_{i=0}^{N} x_{it} \le C, \text{ for each } t \\
& 0 \le x_{it} \le 1
\end{aligned}$$

3) The Probabilistic Approach: The proposed offline algorithm builds upon the greedy algorithm, but it takes a step further by first "shifting" the tasks to less congested area so as to optimally use the resources. Since there is no prior knowledge about congestion at any time slot t, we estimate it using the probability of a task being scheduled at t. Between t^r and t^d, each possible schedule the task will share the same probability, and all of them contribute to the probabilistic congestion. With this estimation, we can smooth out congestion by shifting each task by just once, thus achieving linear complexity.

Now we define our approach in more details. Let n be the number of possible schedules for a task T, where $n = t^d - t^r - \tau^c + 1$, thus each schedule has a probability of $1/n$. Figure 1 shows the probabilistic distribution of two example tasks. Both tasks have $t^r = 0$ and $t^d = 6$, while τ^c equals 3 and 4 respectively. Consider the first task, where one schedule occupies slot 0, thus yielding a probability of 1/4; two schedules occupy slot 2, thus yielding a probability of 2/4, etc. Let $t^t = \min(t^r + \tau^c - 1, t^d - \tau^c)$. The probability can be defined by Equation 1.

$$\Pr(t) = \begin{cases} \frac{t - t^r + 1}{n} & \text{if } t^r \le t < t^t \\ \frac{min(n, \tau^c)}{n} & \text{if } t^t \le t < t^d - \tau^c \\ \frac{t^d - t}{n} & \text{if } t^d - \tau^c \le t < t^d \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Algorithm 1 describes the procedure of probabilistic shifting. The algorithm iteratively finds the least congested slot, and shifts a task in the slot to a position with the minimum average probabilistic congestion. With bounded $t^r - t^d$ and bounded number of time slots, the algorithm execute in $O(N)$ time. After the shifting, we then apply the greedy scheduling algorithm to obtain the final offline decision $D_i(t)$ for each task T_i.

B. Classification

In the classification phase, we train local offload decision $D_i^l(t)$ based on the offline solution $D_i(t)$. The idea is to

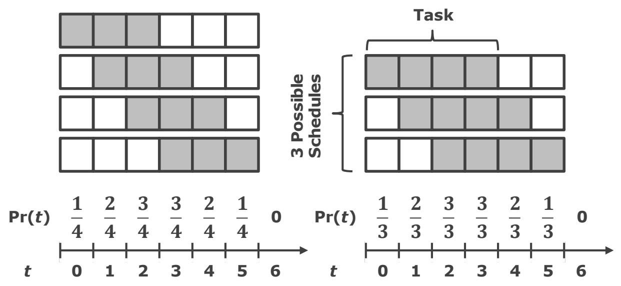

Task
3 Possible Schedules
Pr(t)
$\frac{1}{4}$ $\frac{2}{4}$ $\frac{3}{4}$ $\frac{3}{4}$ $\frac{2}{4}$ $\frac{1}{4}$ 0
$\frac{1}{3}$ $\frac{2}{3}$ $\frac{3}{3}$ $\frac{3}{3}$ $\frac{2}{3}$ $\frac{1}{3}$ 0
t 0 1 2 3 4 5 6

Figure 1: The probabilistic distribution of two example tasks

Algorithm 1 Probabilistic Shifting Algorithm

```
1:  $pc(t) \leftarrow$ probabilistic congestion at time slot $t$
2:  while not all $t$ marked explored do
3:      $t_{min} \leftarrow \underset{t}{\text{argmin}}\ pc(t)$
4:      $\mathbb{T} = \{T_i | t_i^r \le t_{min} < t_i^d, T_i$ is not explored $\}$
5:      if $\mathbb{T} == \emptyset$ then
6:          mark $t_{min}$ as explored
7:      else
8:          select $T_i \in \mathbb{T}$
9:          remove $T_i$ from $pc$
10:         $t_i^{r\prime} \leftarrow \underset{t_i^{r\prime}}{\text{argmin}}\ \frac{1}{t_i^d - t_i^{r\prime}} \left( \sum_{t=t_i^{r\prime}}^{t_i^d} pc(t) + \tau_i^c \cdot 1 \right)$
11:         mark $T_i$ as explored
12:         update $pc(t)$ where $t \in [t_i^{r\prime}, t_i^d)$
13:     end if
14: end while
```

enable local users to be able to assess the quality of tasks, and thus avoid offloading inefficient tasks that are unlikely to be scheduled by the data center. In short, if a similar task is scheduled in the optimal solution, then it's likely to be offloaded to the data center.

1) Classifier: We use *L1 regularized Logistic Regression* classifier for classification. We choose it for its simple representation as a function (as apposed to instances in instance-based learning), which enables cheap update. Further, regression model fits more conceptually with the problem. The underlying relationships between features are continuous, and that no causal relationship (i.e. Bayesian network) exists. Thus we choose logistic regression over other classifiers.

2) Training: The training set is the offline solution $\{(x_i, D_i(t)), i = 1, \ldots, n\}$, where each task T_i has a feature vector x_i, and class label (1: scheduled , 0: not scheduled). The logistic regression model estimates the probability of $\Pr(D = 1|X = x) = 1/(1 + exp(-w^T x))$, where w refers to the parameters of the model. L1 regularization penalizes the model an additional term $\lambda||w||$, where λ is the strength.

Table II shows the list of features we found relevant through regularization. Using different strengths λ_i, different features are selected, and each contribute either positively or negatively to the selection decision (denoted by $+/-$). The most relevant feature is $benefit$, which essentially represents the energy efficiency of the task. Following it is $slack$, defined by $t^d - \tau^r - \tau^t - t^r$, which is another positively weighted feature representing the flexibility of the task. High flexibility grants more waiting time and thus implies higher chance of being scheduled. Our final choice for regularization strength is λ_2, which selects benefit and slack as features.

Table II: Feature Selection

Feature	**Explanation**	λ_1	λ_2	λ_3
τ^c	Cloud Exec. Time			$-$
τ^s	Slack		+	+
b	Benefit	+	+	+

The traffic volume varies at different times of the day, and thus the parameters $w(t)$ should be a function of t. For simplicity, we choose $w(t)$ to be a discrete function, where there is a set of parameters for each predefined period (e.g. 1 hour). We refer the length of the period to be the *granularity* of the model. The logistic regression classifier is also susceptible to bias when the training set is unbalanced (e.g. too many 0 class). Thus in selecting training sets $D_i(t)$ from the offline solution, we use $t = release\ time$ for unscheduled tasks, $t = scheduled\ start\ time$ for scheduled tasks, and carefully keep number balanced.

C. Baselines

In the online scenario, the data center maintains a queue of tasks to be executed. In *Naive* baseline, $D_i^c(t)$ is based on First-Come-First-Serve, while the *Greedy* baseline always schedules the task with the largest benefit. Our *Classified* approach differs from the greedy approach by enabling classified local decision $D_i^l(t)$, instead of offloading everyone.

IV. Evaulation

We simulate a system where the number of tasks is in the scale of 10e5. We run various experiments at a granularity of 10 seconds per time slot over the period of a day, and obtain the results shown in this section.

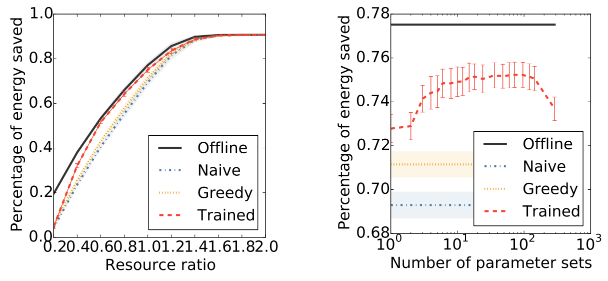
A. System Performance

We conduct 10 runs of experiments using cellular network traces on different days. The number of machines in the data center is set such that around 70% of the tasks can be scheduled. The classification model is built with a granularity of 1 hour. Table III shows the results. We observe that our offline algorithm is almost as good as the upper bound, while the online algorithm performs close to the offline algorithm while outperforming the baselines. The results also demonstrate the stability of our algorithm.

Notice that among the earliest deadline, greedy, and probabilistic algorithm, the number of tasks scheduled decreases yet the energy saved increases. This is due to the

Table III: System Performance

Offline	**Tasks Scheduled /%**	**Energy Saved /%**	**Ratio**
All	100	90.6 ± 0.1	10.63x
Upperbnd	N/A	78.0 ± 1.2	4.545x
Deadline	74.9 ± 1.8	73.3 ± 1.3	3.751x
Greedy	64.9 ± 2.1	76.7 ± 1.3	4.295x
Probabilistic	*63.5 ± 2.2*	*77.2 ± 1.2*	*4.395x*
Online	**Tasks Scheduled /%**	**Energy Saved /%**	**Ratio**
Naive	75.5 ± 1.7	72.0 ± 1.4	3.566x
Greedy	62.7 ± 2.2	73.7 ± 1.4	3.803x
Classified	*60.1 ± 2.2*	*75.7 ± 1.3*	*4.116x*



(a) (b)

Figure 2: (a) shows the performance with different data center resource ratio (b) shows performance with different granularity of training

fact that energy savings of tasks with heavier computation can amortize the fixed components in the offloading overhead, an observation in accordance with [2]. In our power law distributed model, vast majority of tasks are less computationally inexpensive. Thus, scheduling more of the "beneficial" tasks in our proposed approach results in less head counts but better performance.

B. Discussion

1) Data Center Provisioning: We use the ratio of the total amount of resources to the total demand to denote the relative amount resources the data center has. Figure 2(a) demonstrate the energy total energy savings of the system when we provide data center with different amount of resources. Since there are fluctuations in the demand over a day, the resource can be under utilized in most of the time. We observe that a ratio of 1.6 would be sufficient to satisfy the resource requests, a number way below the peak demand. As the amount of resources increases, the increase in energy saved does not grow linearly. It should provide a reference to system designers on how to design the trade-off between energy savings and data center provisioning.

2) Granularity of Classification Model: Figure 2(b) demonstrates performance when using classifier models $w(t)$ with different granularity in t (number different sets of $w(t)$). If the model is shared among large intervals, the model is too coarse-grained and the performance is close to greedy; if intervals are too small, the system will over-fit and thus performance begins to drop again. We see a huge increase from 2 to 3 sets (8 to 12 hour period), and it starts to become steady around using 8 sets (3 hour period).

V. Conclusion

In this paper, we introduce a learning based localized solution to the resource allocation problem in offloading system. We propose a novel time-efficient statistical offline scheduling algorithm that achieves near-optimal solution compared to upper bounds. The classifier-based online algorithm shown performance close to offline algorithms under simulation using realistic workload traces.

Acknowledgements

This work was supported in part by the NSF under award CNS-0958369, award CNS-1059435, and award CCF-0926127, and by Samsung under award GRO-20130123.

References

[1] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: making smartphones last longer with code offload," in *Proceedings of the 8th International Conference on Mobile Systems, Applications, and Services, MobiSys*, 2010, pp. 49–62.

[2] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *European Conference on Computer Systems, Proceedings of the Sixth European conference on Computer systems, EuroSys*, 2011, pp. 301–314.

[3] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proceedings of the IEEE INFOCOM 2012*, 2012, pp. 945–953.

[4] S. Kang, J. Lee, H. Jang, H. Lee, Y. Lee, S. Park, T. Park, and J. Song, "Seemon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments," in *Proceedings of the 6th International Conference on Mobile Systems, Applications, and Services, MobiSys*, 2008, pp. 267–280.

[5] J. H. Ahnn and M. Potkonjak, "mhealthmon: Toward energy-efficient and distributed mobile health monitoring using parallel offloading," *J. Medical Systems*, vol. 37, no. 5, p. 9957, 2013.

[6] L. A. Barroso, J. Clidaras, and U. Hölzle, *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines, Second Edition*, ser. Synthesis Lectures on Computer Architecture. Morgan & Claypool Publishers, 2013.

[7] K. Kumar and Y. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *IEEE Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[8] J. Wilkes, "More Google cluster data," Google research blog, Nov. 2011, posted at http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.

[9] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.

[10] A. Bar-Noy, S. Guha, J. Naor, and B. Schieber, "Approximating the throughput of multiple machines in real-time scheduling," *SIAM J. Comput.*, vol. 31, no. 2, pp. 331–352, 2001.