Utility-based Uploading Strategy in Cloud Scenarios

Ziqi Wan and Jie Wu

Temple University, Philadelphia, PA, USA
Introduction

• The Cloud provider
  – We care customers.
  – We’d like to **make money**!

• How to make money?
  – Profit = (Unit Price – Unit Cost) * Number of Users
  – Lower cost leads lower price and more users.
  – **Lower data center’s cost**!

• User
  – The task should be finished quickly.
Cost is HUGE!

- Data center burns money!
  - Annual cost to operate a single data center runs between $10 million and $25 million

- Use less machines if possible!
  - Lower the running power cost!
  - Lower cooling power cost!
  - Don’t even need to buy them!
Problems

• How to save machine number?
  – Run more jobs in less machines
  – However, users need to wait for a longer time. (Customers might complain, and quit the contract.)

• How to save time?
  – Just rent more machines for a single job

• With a limited budget.
  – If you want to go quickly, go alone. If you want to go far, go together.

• Key issue:
  – Should I wait?
  – How long should I wait?
Simplification

• Balance customers’ satisfaction and data center cost

• Simplification:
  – Task Finishing Time $\rightarrow$ Customers’ Satisfaction
  – Running Machines Cost $\rightarrow$ Data Center Cost

Again, balancing Time and Money!
Outline

1. Introduction
2. Problem Formulation
3. Models
4. Analysis
5. Simulation Results
6. Conclusions and the Future Work
Utility-based VM Model

- Utility Cost Model

\[ U_i = B_i - p_i - b \times t_i \]

- Then we just need to minimize the utility cost to maximize the profit for cloud providers.
Utility-based VM Model (Cont’d)

• More VMs assign to a job will make this job run faster.
• However, the processing speed not linearly increase with the number of machines.

![Graphs showing average time and normalized machine time vs. number of machines.](image)

(a) Average Time  (b) Normalized Machine Time

Fig. 3. Word Count
The processing speed not linearly increase with the number of machines.

Fig. 4. Pentomino

Fig. 5. TeraSort
Analysis

• In some cases, users only care about the time, and pay little attention to the rent price.
  – minimize the time cost first, then consider minimizing the machine rent price.

• In some cases, users only care about the Price
  – minimize the rent price first, then consider minimizing the machine time cost.

• A simple policy-shifting algorithm
  – In case, we don’t know which one is more important
More Analysis

• We want to maximize the utility directly.
  – We provide a greedy algorithm to make a balance between performance and time complexity.

```
Algorithm 6 Group Utility – greedy size

Input: Workloads of all jobs, total number of machines, and speedup property of machines;

1: h=1
2: while There is a jobs not in groups $g_1, g_2, ..., g_h$ do
3:   for $g_h$ from 1 to maximum number of machines $M$ do
4:     while $N > \sum_{j=1}^{h} g_j$ do
5:       Set Number of jobs in $g_{h+1}$ as $g_h$
6:      Compute total $U = \sum (B_i - Uc_i)$ of the all group;
7:      $h = h + 1$;
8:   Compare and find out the best number of jobs for $g_h$.
9:   Schedule jobs in groups with that number.
```
Simulation

• Simulation setting
  – 8 Dell R210 Servers.
    • dual core Intel Celeron processor.
    • 4 GB of RAM
  – Cisco small business 300 Series Managed Switch
  – Tasks (three common applications in the Hadoop cloud framework)
    • Word Count, Pentomino, and TeraSort
Simulation

• Algorithms
  – Time-first algorithm
  – Price-first algorithm
  – Utility-single algorithm
  – Utility-greedy algorithm
Simulation (cont’d)

Simulation Results of 4 Algorithms for Word Count
Outline

1. Introduction
2. Problem Formulation
3. Models
4. Analysis
5. Simulation Results
6. Conclusions and the Future Work
Conclusions

• We consider the design and analysis utility-based scheduler in the cloud environment. Unlike all existing works, we propose the notion of the utility for the Virtual Machine management.
• The model presented here opens the door for an in-depth study of how to schedule in the presence of phase overlapping. There are a wide variety of open questions remaining with respect to the design of algorithms that minimize response time
Thank you!

Questions?

ziqi.wan@temple.edu