Optimal Migration Contracts in Virtual Networks: Pay-as-You-Come vs Pay-as-You-Go

Xinhui Hu*, Stefan Schmid’, Andrea Richa* and Anja Feldmann’

*Arizona State University
’Telekom Innovation Laboratories & TU Berlin
Outline

- Motivation
- Related Work
- System Model and Problem Formulation
- Algorithm
- Performance Evaluation
- Conclusion
Motivation

Virtual Networking of Cloud Resources
Research Challenges

- When and where to migrate a service?

- Offline/online algorithm!
  - Offline: every day the same
  - Online: no knowledge of the future requests

- Economical dimension:
  - Migration comes at costs: contracts!
Our Perspective

- Migration contracts
  - Contracts with more bandwidth or longer duration are cheaper! (discounts)
  - Which contract (bandwidth, duration) to buy?

- Objective
  - Find the optimal migration contracts in virtual networks for two pricing models:
    - Pay-as-You-Come: “pay in advance even if not needed”
    - Pay-as-You-Go: “pay for what you use only”
Contribution

- We present *two optimal offline algorithms* for migration contracts in virtual networks for Pay-as-You-Come and Pay-as-You-Go pricing models (PAYC and PAYG)

- We present *two online algorithms* for Pay-as-You-Come and Pay-as-You-Go pricing models (ONC and ONG)
Outline

- Motivation
- Related Work
- System Model and Problem Formulation
- Algorithm
- Performance Evaluation
- Conclusion
Related Work

- Online Model but without Economics:
  

- CloudNet Prototype (with NTT DoCoMo):
  

- Much more literature on economical aspects of cloud pricing
Outline

- Motivation
- Related Work
- System Model and Problem Formulation
- Algorithm
- Performance Evaluation
- Conclusion
Cost Model

- **Access cost**: latency associated with satisfying request remotely
- **Migration cost**: cost of migrating server from current location to location of request (cost = service interruption time, depends on bandwidth)
- **Contract cost**: cost of buying/renting resources (different discounts are provided for different contract bandwidths and durations)
Pricing Model

- **Pay-as-You-Come**: pay for the resource in advance, before resource is utilized
  - if not used, it’s your fault! (like MPLS 😊)

- **Pay-as-You-Go**: pay for the resource after resource is utilized
  - only if actually utilized (like EC2)
Data Model

- **Contract durations**: \( \tilde{D} = \{d_1, d_2, ..., d_k\}, d_1 \leq d_2 \leq ... \leq d_k \)
- **Contract bandwidths**: \( \tilde{B} = \{b_1, b_2, ..., b_q\}, b_1 \leq b_2 \leq ... \leq b_q \)
- **Discount function**: \( f \)
  - Linear
  - Non-linear: e.g., \( \sqrt{\cdot} \), \( \log \), ...
  - For example, a twice as long contract may cost only 50% more, and doubling the reserved bandwidth may cost only 30% more.
- **Request sequence**: \( \langle r_1, t_1 \rangle, \langle r_2, t_2 \rangle, ... \)
  - \( \langle r_i, t_i \rangle \) represents the \( i^{th} \) request from \( r_i \) at time \( t_i \)
- **Two sites**: L (e.g., USA) and R (e.g., Japan)
An example

d=7, b=50Mbps
Problem Formulation

Given requests \(<r_1, r_2, ..., r_n>\) at respective times \(<t_1, t_2, ..., t_n>\), we aim to minimize

\[
\text{Cost} = \text{AccCost} + \text{MigCost} + \text{ConCost}
\]

where

- **Access cost:**
  \[
  \text{AccCost} = \sum_i D[r_i, s_i]
  \]

- **Migration cost:**
  \[
  \text{MigCost} = \sum_i S \cdot D[s_{i-1}, s_i]/b_i
  \]

- **Contract cost:**
  - Pay-as-You-Come:
    \[
    \text{ConCost} = \sum_i f(d_i, b_i)
    \]
  - Pay-as-You-Go:
    \[
    \text{ConCost} = f(\mu, b_i)
    \]
Outline

- Motivation
- Related Work
- System Model and Problem Formulation
- Algorithm
- Performance Evaluation
- Conclusion
Main Results

- **Optimal algorithms** (based on *novel dynamic programming* approaches)
  - PAYC for Pay-as-You-Come pricing model in $O(n^2(n + kq))$
  - PAYG for Pay-as-You-Go pricing model in $O(qn^3)$

- **Online algorithms**
  - ONC for Pay-as-You-Come pricing model
  - ONC for Pay-as-You-Go pricing model

- **Experimental evaluation**
## Data structure for PAYC

<table>
<thead>
<tr>
<th>Dynamic Progr. Matrix</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{n \times n \times 4}$</td>
<td>Total cost matrix in PAYC, an entry $C[i,j,k]$, where $i,j \in [n]$ and $k \in {(s, s')</td>
</tr>
<tr>
<td>$(AM_m)_{n \times n \times 4}$</td>
<td>Combined access cost and migration cost matrix for bandwidth $b_m$, $AM_m[i, j, (s, s')]$ stores the combined access and migration costs for the best migration strategy that satisfies the sequence of requests from $r_i$ to $r_j$</td>
</tr>
</tbody>
</table>
PAYC: Satisfying one request

- service request from node \( s \) at time \( t \), and server size \( S \)
- If the server is also located at \( s \) then \textbf{no cost}
- Else, either pay
  - access cost or
  - pay server migration cost \( S/bm \) and \( d_1 \)-day contract cost \( f(d_1,bm) \), for some bandwidth \( bm \) and smallest contract duration \( d_1 \)
PAYC: Satisfying multiple requests

multiple requests from time $t_i$ to time $t_j$

Split the contract at $u_{th}$ request

\[
\begin{align*}
\text{Buy a long contract } d_v \text{ to cover the interval, where } v \in [k] \\
\end{align*}
\]
Algorithm 1 Algorithm PAYC

**Input:** Requests \(<r_1, r_2, \ldots, r_n>\) at respective times \(<t_1, t_2, \ldots, t_n>\).

**Output:** Minimum cost.

1. for \(i = 1\) to \(n\) do
2. for all pairs \((s, s') \in \{L, R\}^2\) do
3. for \(m = 1\) to \(q\) do
4. \(AM_m[i, i, (s, s')] \leftarrow D[s', r_i] + S \cdot D[s, s']/b_m\)
5. \(C[i, i, (s, s')] \leftarrow \min_{1 \leq m \leq q}\{AM_m[i, i, (s, s')] + f(d_1 \ast D[s, s'], b_m)\}\)
6. for \(l = 2\) to \(n\) do
7. for \(i = 1\) to \(n - l + 1\) and pairs \((s, s') \in \{L, R\}^2\) do
8. \(j \leftarrow i + l - 1\)
9. \(C[i, j, (s, s')] \leftarrow \min_{i \leq u < j; s'' \in \{L, R\}}\{C[i, u, (s, s'')] + C[u + 1, j, (s'', s')]\}\)
10. if \(d_{v-1} < t_j - t_i + 1 \leq d_v\), for some \(v = \{1, \ldots, k\}\) then
11. for \(m = 1\) to \(q\) do
12. \(AM_m[i, j, (s, s')] \leftarrow \min_{s'' \in \{L, R\}}\{AM_m[i, i, (s, s'')] + AM_m[i + 1, j, (s'', s')]\}\)
13. if \(C[i, j, (s, s')] > \min_{1 \leq m \leq q}\{AM_m[i, j, (s, s')] + f(d_v, b_m)\}\) then
14. \(C[i, j, (s, s')] \leftarrow \min_{1 \leq m \leq q}\{AM_m[i, j, (s, s')] + f(d_v, b_m)\}\)
15. return \(\min_{s_{\text{final}} \in \{L, R\}} C[1, n, (s_{\text{init}}, s_{\text{final}})]\)
## Data structure for PAYG

<table>
<thead>
<tr>
<th>Dyn. Progr. Matrix</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_{n \times n \times 4}$</td>
<td>Total cost matrices in PAYC, an entry $C[i,j,k]$, where $i,j \in [n]$ and $k \in {(s, s')</td>
</tr>
<tr>
<td>$(A_m)_{n \times n \times 4}$</td>
<td>Access cost matrix for bandwidth $b_m$</td>
</tr>
<tr>
<td>$(N_m)_{n \times n \times 4}$</td>
<td>Number of migrations matrix for bandwidth $b_m$</td>
</tr>
</tbody>
</table>
PAYG: One request

Similar dynamic programming approach as PAYC is used in PAYG. The main difference is how to update the cost for multiple requests.

Initialization:

Access cost:

\[ A_m[i, i, (s, s')] \leftarrow D[s', r_i] \quad m \in [q] \]

Migration cost:

\[ N_m[i, i, (s, s')] \leftarrow D[s, s'] \]

Total cost:

\[ C_m[i, i, (s, s')] \leftarrow A_m[i, i, (s, s')] + S \cdot N_m[i, i, (s, s')]/b_m + f(D[s, s'], b_m) \]
PAYG: Multiple requests

multiple requests from time $t_i$ to time $t_j$

Split the contract at $u_{th}$ request

\[
\begin{align*}
& C_m[i, j, (s, s')] \leftarrow \min_{i \leq u < j; s'' \in \{L, R\}} \{A_m[i, u, (s, s'')] + A_m[u + 1, j, (s'', s')] + \sum_{s'''} (N_m[i, u, (s, s''')] + N_m[u + 1, j, (s'', s''')] / b_m + f((N_m[i, u, (s, s''')] + N_m[u + 1, j, (s'', s''')], b_m)) \}
\end{align*}
\]

The optimal result is given by:

\[
\min_{s_{\text{final}} \in \{L, R\}} \min_{1 \leq m \leq q} C_m[1, n, (s_{\text{init}}, s_{\text{final}})]
\]
Algorithm 2 Algorithm PAYG

Input: Requests \(<r_1, r_2, ..., r_n>\) at respective times \(<t_1, t_2, ..., t_n>\).

Output: Minimum Cost.

1: for \(i = 1\) to \(n\) do
2:     for all pairs \((s, s') \in \{L, R\}^2\) and \(1 \leq m \leq q\) do
3:         \(A_m[i, i, (s, s')] \leftarrow D[s', r_i]\)
4:         \(N_m[i, i, (s, s')] \leftarrow D[s, s']\)
5:         \(C_m[i, i, (s, s')] \leftarrow A_m[i, i, (s, s')] + S \cdot N_m[i, i, (s, s')] / b_m + f(D[s, s'], b_m)\)
6:     for \(l = 2\) to \(n\) do
7:         for \(i = 1\) to \(n - l + 1\) do
8:             \(j \leftarrow i + l - 1\)
9:         for all pairs \((s, s') \in \{L, R\}^2\) and \(1 \leq m \leq q\) do
10:         \(C_m[i, j, (s, s')] \leftarrow \min_{i \leq u < j; s'' \in \{L, R\}} \{A_m[i, u, (s, s'')] + A_m[u + 1, j, (s'', s')] + S \cdot (N_m[i, u, (s, s'')] + N_m[u + 1, j, (s'', s')]) / b_m + f((N_m[i, u, (s, s'')] + N_m[u + 1, j, (s'', s')]), b_m)\}
11:     Let \((u, s'')\) be the parameter and location of request \(r_u\) at \(t_u\) that minimized Line 10.
12:     \(A_m[i, j, (s, s')] \leftarrow A_m[i, u, (s, s'')] + A_m[u + 1, j, (s'', s')]\)
13:     \(N_m[i, j, (s, s')] \leftarrow N_m[i, u, (s, s'')] + N_m[u + 1, j, (s'', s')]\)
14: return \(\min_{s_{\text{final}} \in \{L, R\}, 1 \leq m \leq q} C_m[1, n, (s_{\text{init}}, s_{\text{final}})]\)
Online algorithm: ONC

- We do an amortization by migrating only when the access cost (C) exceeds the migration cost.
- If no contract is available for current migration, ONC checks if a longer contract would have been better for the past requests.
- Otherwise, ONC checks whether a shorter contract should be chosen.
Online algorithm: ONG

- A counter $C_1$ records the number of the migrations performed so far while $C_2$ denotes the total access costs.
- If the access cost $C_2$ reaches the migration cost plus marginal migration contract costs (i.e., $f(C_1+1, b)-f(C_1, b)$, for bandwidth $b$), ONG migrates the server, increments counter $C_1$, and resets counter $C_2$. 
Outline

- Motivation
- Related Work
- System Model and Problem Formulation
- Algorithm
- Performance Evaluation
- Conclusion
Simulation Setting

- **Duration set:** $D = \{1, 30, 60, 100\}$
- **Bandwidth set (Mbps):** $B = \{50, 100\}$
- **Server size:** $S = 250M$
- **Unit access cost:** 5
- **Request number:** $n = 1500$ requests
- **Discount function:**
  
  \[
  f_{\text{lin}} = 1.5 \log \frac{d_i + b_j}{50} - 1 \cdot 6
  \]
  
  \[
  f_{\text{sqrt}} = \sqrt{d_i b_j / 50} \cdot 6
  \]
  
  \[
  f_{\log} = \log(d_i b_j / 50) \cdot 6
  \]
Simulation Setting

- Request model:
  - Requests alternate between the two virtual sites
  - Markov process: stay at current site according to an exponential distribution with parameter $\lambda$, then change with probability $p$
Cost distribution for PAYC and PAYG
### Table 1. Distribution of purchased contracts (discount function $f_{lim}$).

<table>
<thead>
<tr>
<th>Dur-Bw</th>
<th>$\lambda$</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-50</td>
<td>11.2</td>
<td>8</td>
<td>15.4</td>
<td>13.8</td>
<td>18.4</td>
<td>39.2</td>
<td></td>
</tr>
<tr>
<td>60-50</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2.4</td>
<td>0.8</td>
<td></td>
</tr>
<tr>
<td>60-100</td>
<td>1.4</td>
<td>2</td>
<td>1.4</td>
<td>2.8</td>
<td>1</td>
<td>0.4</td>
<td></td>
</tr>
<tr>
<td>100-50</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.6</td>
<td>2</td>
<td>5.4</td>
<td></td>
</tr>
<tr>
<td>100-100</td>
<td>11</td>
<td>11</td>
<td>11.2</td>
<td>10</td>
<td>7.6</td>
<td>3.4</td>
<td></td>
</tr>
</tbody>
</table>

### Table 2. Number of migrations for each contract (discount function $f_{lim}$).

<table>
<thead>
<tr>
<th>Dur-Bw</th>
<th>$\lambda$</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-50</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>60-50</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>8.5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>60-100</td>
<td>17.67</td>
<td>14</td>
<td>13.5</td>
<td>11.5</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>100-50</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>13</td>
<td>13</td>
<td>12.57</td>
<td></td>
</tr>
<tr>
<td>100-100</td>
<td>27.33</td>
<td>23.58</td>
<td>19.45</td>
<td>17.33</td>
<td>15</td>
<td>14.5</td>
<td></td>
</tr>
</tbody>
</table>
Competitive ratio

(a) Competitive ratio for ONC.

(b) Competitive ratio for ONG.
Outline

- Motivation
- Related Work
- System Model and Problem Formulation
- Algorithm
- Performance Evaluation
- Conclusion
Conclusion and Future Work

Conclusion

- We have studied *online migration in virtual networks from an economical perspective and in two pricing models: Pay-as-You-Come and Pay-as-You-Go*
- We have presented *optimal algorithms for each pricing model*
- We have discussed *online algorithms for each pricing model*

Future Work

- Extend to live migration
- Extend to more complicated virtual networks (more than two sites)
THANK YOU!

Questions ?
Backup slides
Optimal algorithm: PAYC

One request: request from \( r_i \) at time \( t_i \)

\[
\begin{align*}
\text{Pay access cost if request is from remote location (} r_i \neq s' \text{)} \quad & \\
\text{If } s \neq s' \text{ then, for bandwidth } b_m, \text{ pay migration cost } S/b_m \text{ and } d_1 \text{-day contract cost } f(d_1, b_m)
\end{align*}
\]

Server location \( s' \)

Access+Migration cost:

\[
AM_m[i, i, (s, s')] \leftarrow D[s', r_i] + S \cdot D[s, s']/b_m \quad m \in [q]
\]

Total cost:

\[
C[i, i, (s, s')] \leftarrow \min_{1 \leq m \leq q} \{AM_m[i, i, (s, s')] + f(d_1 * D[s, s'], b_m)\}
\]
Cost distribution for PAYC

Observation

- The total cost and the access cost decrease for larger lambda
- The migration and contract stay much more stable

Reason

- Requests originating from one site for longer time periods render it worthwhile to migrate and buy longer contracts
Cost distribution for PAYG

- **Observation:**
  - The total cost is lower than that of PAYC
  - The migrations constitute a larger share of the overall costs

- **Reason:**
  - The contract cost is given by the number of migrations and the amount of leased bandwidth under the discount function
Number of migrations and effect of discount function

- The experiments are derived under $f_{\text{lin}}$ discount function