QoS-Aware Service Selection in Geographically Distributed Clouds

Xin Li*,†, Jie Wu†, and Sanglu Lu*

*State Key Laboratory for Novel Software Technology, Nanjing University, China
†Department of Computer and Information Science, Temple University, USA
Outline

- Background and Motivation
- Problem Statement
- Our Approach
- Evaluation
Background

- Cloud service
  - More and more services are accessible with the growth of cloud computing.
  - Functional simplicity
    - It is functional limited for a single service.
    - How can we effectively utilize the plentiful services?

- Service composition
  - An effective way to utilize the plentiful services, and compensates for the simplicity of single service.
Service Composition

- To accomplish an integrated task, many basic processing units are composed to achieve the complicated job.
- Service (functional) component
  - The basic processing unit

An example of service composition for multimedia delivery

- $f_0$: media source
- $f_1$: media logo
- $f_2$: compression
- $f_3$: subtitle embedding
- $f_4$: transcoding
A functional graph is defined as $FG = < F, E, \lambda, K >$

- $F$: the set of functional components
- $E$: the set of functional edges between components
- $\lambda$: the number of components
- $K$: the number of functional paths

Functional paths:
- $F_1 F_2 F_3 F_4 F_\lambda$
- $F_1 F_2 F_4 F_\lambda$
- $F_1 F_2 F_3 F_4 F_\lambda$
- $F_1 F_2 F_3 F_5 F_\lambda$
Motivation

- The opportunity
  - For service provider: rent the existing single services from various clouds, and provide some complex integrated services to users.

- The challenge
  - Functional equivalent service instances
    - Service instance: an implementation of a service component
    - Same function but different features, e.g. execution time
  - Cost limitation
  - The quality of service instances
The Problem

- Problem description
  - Select a set of service instances such that the composed service offers the best quality under given functional graph.
  - An instantiation for the functional graph
The Problem

cloud (instance)

user

We are here!
Instantiation of Functional Graph

Abstract level

Concrete level

Functional component
Functional edge

Service instance
Network link
Data flow

Latency
## Abstract Level & Concrete Level

### Notation Contradistinction

<table>
<thead>
<tr>
<th>Abstract Level</th>
<th>Concrete Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>functional component</td>
<td>service instance</td>
</tr>
<tr>
<td>$F_i$, $0 \leq i \leq \lambda$</td>
<td>$F_i^j$, $j = 1, 2, \ldots, \theta(i)$</td>
</tr>
<tr>
<td>functional path</td>
<td>data flow</td>
</tr>
<tr>
<td>$P_k$, $1 \leq k \leq K$</td>
<td>$\omega_k$</td>
</tr>
<tr>
<td>functional edge</td>
<td>network link</td>
</tr>
<tr>
<td>$E_{ij}$</td>
<td>$L_{ij}$</td>
</tr>
</tbody>
</table>

**Selection function**

$\pi$ and $\pi'$:

$\pi(F_i) = F_i^j, \pi(E_{ij}) = L_{ij}, \pi'(\omega_k) = P_k$
Quality of Service Instance Set

- How to judge the quality of a set of instances?
  - Quality of an instance set
  - QoS (quality of service)
    - Objective: best quality
      - The minimal expected response time for all users
    - Network model
      - How to measure the network latency between two locations?
      - Network Coordinate System (NCS)
Network Coordinate System (NCS)

- Network latency
  - It is infeasible to gather the network latency.
    - Due to the large amount of measure traffic.
  - NCS is widely used to estimate the network latency.
  - Latency varies a lot depending on the location.
  - 2-dimensional coordinate system
  - Communication latency: $c(l_1, l_2) = \alpha + \beta \cdot dist(l_1, l_2)$
    - $dist(l_1, l_2)$: Euclidean distance between the two locations $l_1$ and $l_2$. 
Rethink the Problem

Objective

- Minimize the response time
  - Network latency
  - Execution time
- However, the response time varies
  - User location
    - For users located on different locations, the network latency varies.
  - Functional path selection
    - Given functional graph, there will be multiple functional paths, the response time varies as the path selection.
    - For some user, there is a probability distribution to determine which path will be selected, according to the context.
Delay of Data Flow

A data flow is an instantiation of a functional path.

The delay of a data flow consists of two parts

- Network latency
  - For all of the network links
  - (functional edges)

- Execution time
  - For all service instances
  - (functional component)
Case for One Data Flow

\[ D(\omega) = \sum_{i=1}^{\lambda} T(F_i^\omega) \ast Z(i, \pi'(\omega)) + \sum_{i \leq i, j \leq \lambda} c(F_i^\omega, F_j^\omega) \ast Z(i, j, \pi'(\omega)) \]

\(\lambda\) : the number of functional components for given functional graph

\(F_i^\omega\) : the instance of component \(F_i\) in the flow \(\omega\)

\(T(F_i^\omega)\) : execution time of the instance \(F_i^\omega\)

\(\pi'(\omega)\) : the functional path corresponds to the flow \(\omega\)

\(c(x, y)\) : the network latency between instance \(x\) and \(y\).

\[ Z(i, k) = \begin{cases} 1, & P_k \text{ contains } F_i; \\ 0, & \text{otherwise}. \end{cases} \]

\[ Z(i, j, k) = \begin{cases} 1, & P_k \text{ contains } E_{ij}; \\ 0, & \text{otherwise}. \end{cases} \]
Response Time (1)

- Response time for given data flow $\omega$ and user $u$ (user location)
  - $R(u, \omega) = D(\omega) + c(u, F^\omega_1) + c(F^\omega_\lambda, u)$
    - Total response time for user $u$
  - Data flow delay: 255ms
    - Network latency: $(40+35+50)$
    - Execution time: $(30+25+35+40)$
  - Response time: 320ms
    - $255+45+20$
Response Time (2)

- Response time for given function path $P_k$ and user $u$
  - $R(u, k) = R(u, \omega^\text{opt}_k) \leq R(u, \omega^j_k)(1 \leq j \leq J_k)$
  - $\omega^j_k(1 \leq j \leq J_k)$: the available flows of path $P_k$
  - $J_k = \prod_{i=1}^{\lambda} N(F_i)^{Z(i,k)}$
  - $N(F_i)$: the number of selected instance of component $F_i$

The data flow with minimal delay will be selected to execute the service for given user.
Response Time (3)

- Response time (expected) for given functional graph $FG$ and user $u$
  - $R(u) = \sum_{k=1}^{K} p(P_k) \times R(u, k)$
  - $p(P_k)$: the probability of path $k$ will be selected.
  - For given functional graph, there is an optimal set of service instances for each user. But the optimal set of each user varies.
Quality of Service Instance Set

- Response time is based on the service instances that can be selected.
  - The data flows are based on the instance set $S$.
- The quality for given service instance set $S$.
  - $Q(S) = \frac{1}{\mu} \sum_{u=1}^{\mu} R(u)$
  - $\mu$: the number of users
    - It can be the representative user, and reflects the user distribution.

- **Our problem again**
  - Select a set of service instances set $S$ such that $Q(S)$ is minimal.
Algorithms

- **Simple case**
  - One instance for each functional component

- **General case**
  - Multiple instances for each functional component
  - Limitation for the total number of instances
Simple Case

- For given functional graph $FG = \langle F, E, \lambda, K \rangle$, only one service instance could be selected for each functional component.
  - $\forall i, \ N(F_i) = 1, \sum_i N(F_i) = \lambda$

- Basic idea
  - From initial component to the terminal component, the flow with the minimal latency should be selected.
    - The idea of shortest path algorithm
  - Initial component
  - Terminal component
Simple Case - Algorithm

- Initial (Terminal) component selection
  - Selected in greedy manner, according to the user distribution.
  - The facility location problem.

- Source – Destination (flow selection)
  - From initial component to terminal component
  - The shortest part problem
    - Each instance has an executing time
General Case

- There may be multiple service instances for each component, but the total number of instances is limited.
  - $\forall i, N(F_i) \geq 1$
  - $\sum_i N(F_i) \leq \gamma$
  - For the users, it is more likely to achieve the optimal instance set for her, since more service instances can be selected.
  - Cost limitation
    - The service provider cannot rent too many instances.
General Case - Algorithm

- Basic idea
  - Vote
    - For each user, there is a instance set, that best satisfies the user.
    - Each user declares her preference for the service selection.
    - For each service instance, each user marks a score for it.
    - The score is based on shortest path selection.
  - Select
    - For each component, the instance with the highest score will be selected.
    - Then, other instances are sorted in descending order, and select the first $\gamma - \lambda$ instances.
Evaluation – Simulation Setup

- **Settings**
  - Function graph
    - Random probability
  - NCS
    - $\alpha = 1, \beta = 50$
  - User/Instance location
    - Two-dimensional coordinate space: $[0,1] \times [0,1]$
  - The number of candidate instances ($\theta$) is the same for all functional components.
  - Baseline
    - Random algorithm
Result – Simple Case

- Impact of $\mu$

![Graph showing the impact of $\mu$ and $\theta$ on the ratio of users for different values of $\lambda$.]

- Impact of $\theta$.
- Case when $\mu = 1$.
- Case when $\theta = 2$.
- Impact of $\mu$. 

2013/8/5
Result – General Case

- Impact of $\gamma$

More selected instances provide better quality (less delay). But, when $\gamma - \lambda$ comes to some threshold, the ratio tends to be steady, because the service instances with high score have yet to be selected. The new selected are useless in improve the quality.

The voting algorithm and the random algorithm have the same result, when all instances are selected. (red line)
Conclusion

- **Service selection**
  - A set of service instances with guaranteed quality.

- **Quality of selected instances**
  - Network latency
  - Data flow delay
  - Response time

- **Algorithm**
  - Simple case
  - General case
QoS-Aware Service Selection in Geographically Distributed Clouds

Thanks!

Xin Li*,†, Jie Wu†, and Sanglu Lu*

*State Key Laboratory for Novel Software Technology, Nanjing University, China
†Department of Computer and Information Science, Temple University, USA