QoS-Aware Service Selection in Geographically Distributed Clouds

Xin Li [1]
Jie Wu [2]
Sanglu Lu [1]

1. Nanjing University
2. Temple University

Outline

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

Background

Cloud Service

- More and more services can be accessible with the growth of cloud computing.
  - All over the world
- There are many services with equivalent functions but various quality, e.g. execution time.
- Service composition is an effective way to utilize the plenitude of services.

Motivation

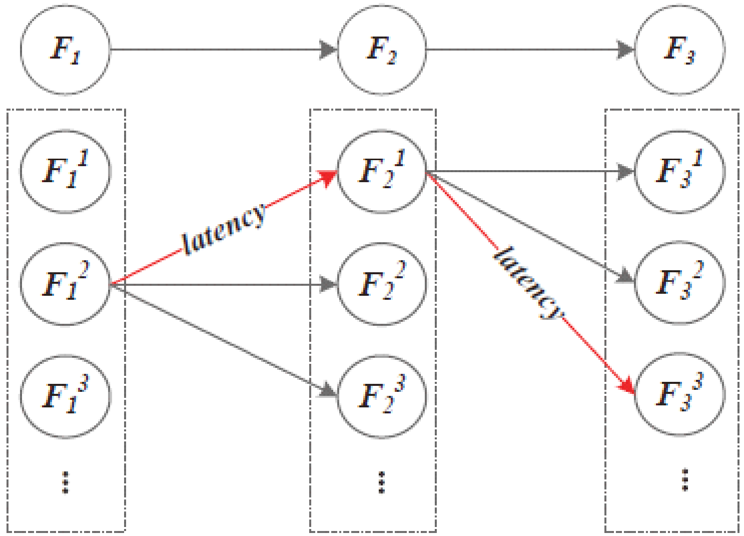
An opportunity

- This vision provides a new opportunity.
- Satisfy the diverse demands of users via service composition based on the cloud services.
- Provide the best QoS for the users.
  - Minimal latency

The problem

- How to select the optimal service set when many functional equivalent servers exist?
- The total number of service instance is limited due to the constraint of cost.

Motivation Example

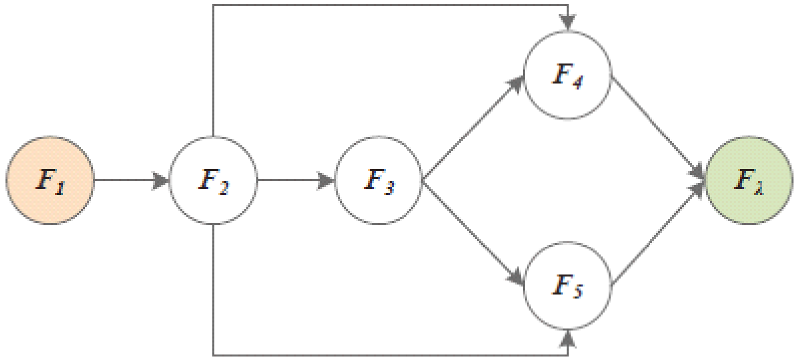


An example of service composition with 3 basic simple services.

Which services should be selected as composition components?

Preliminaries

❖ Function Graph



❖ Initialization (abstract level & concrete level)

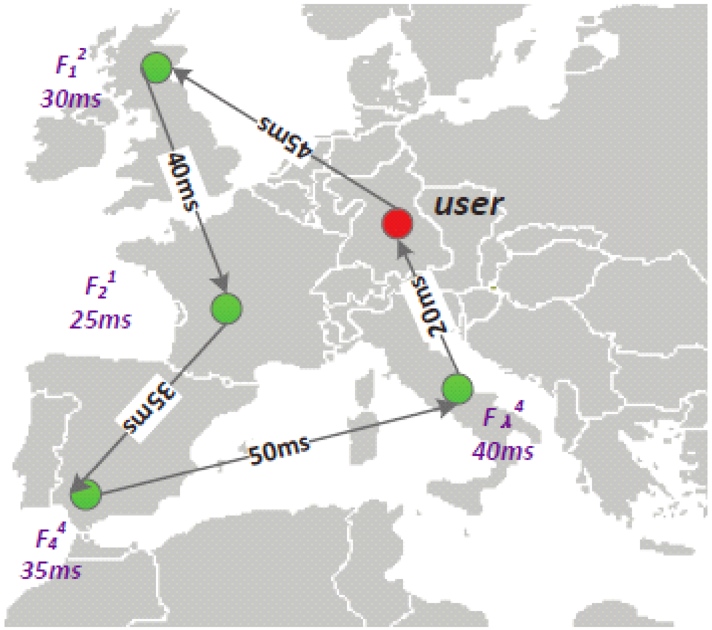
- Select service instances
  - *for the abstract functional component .*

Preliminaries (Cont.)

❖ Network Model

- NCS (network coordinate system)

❖ Data Flow



Data Flow

- Total response time for user and data flow .

- Delay of a data flow .

Total Quality

- Latency for some user .

- Quality for the selected service set
- Factor to measure the total QoS.

Problem Statement

Problem Formalization

$$
\begin{aligned}
min. \quad & \frac{1}{\mu}\sum_{u=1}^{\mu} R(u) \\
s.t. \quad & \sum_{i=1}^{\lambda} N(F_i) \le \gamma \\
& 1 \le N(F_i) \le |I_i|, \forall 1 \le i \le \lambda
\end{aligned}
$$

Hardness

- NP-hard

Algorithm – Simple Case

- **Only one service instance for each component.**
- **Basic idea**
  - Select the instance for initial and terminal component.
  - Shortest path

Algorithm 1 Selection Algorithm

Input: the user set U, service instance set I, functional graph $FG = < F, E, \lambda, K >$

Output: service instance set $\mathcal{S}$, where $|S| = \lambda, \forall r, t \in S, \pi'(r) \neq \pi'(t)$.

```
S ← ∅
π(F_1) = facilityLoc(U, I_1)
π(F_λ) = facilityLoc(U, I_λ)
for k = 1; k ≤ K; k = k + 1 do
    S = S ∪ shortestPath(k, π(F_1), π(F_λ))
S = combine(S)
return S
```

Algorithm – General Case

- **There may be multiple instances for each component, but the total number of instances in limited.**
- **Basic idea**
  - Voting: each user declares her preference for the service selection.

Algorithm 3 *voting(u,k)*

Input: service instance set I, user u, and functional path P_k.
Output: service instance set $\mathcal{S}$

$\mathcal{S} \leftarrow shortestPath(I, u, k)$
for $\forall s \in \mathcal{S}$ **do**
    $s.score \leftarrow s.score + \wp(P_k)$

General Case (Cont.)

❖ Basic idea

- Voting
- Selection: sort the instances according to their scores, which is the results of voting.

Algorithm 2 Voting Algorithm

Input: the user set U, service instance set I, functional graph $FG =< F, E, \lambda, K >$, instance number limitation γ

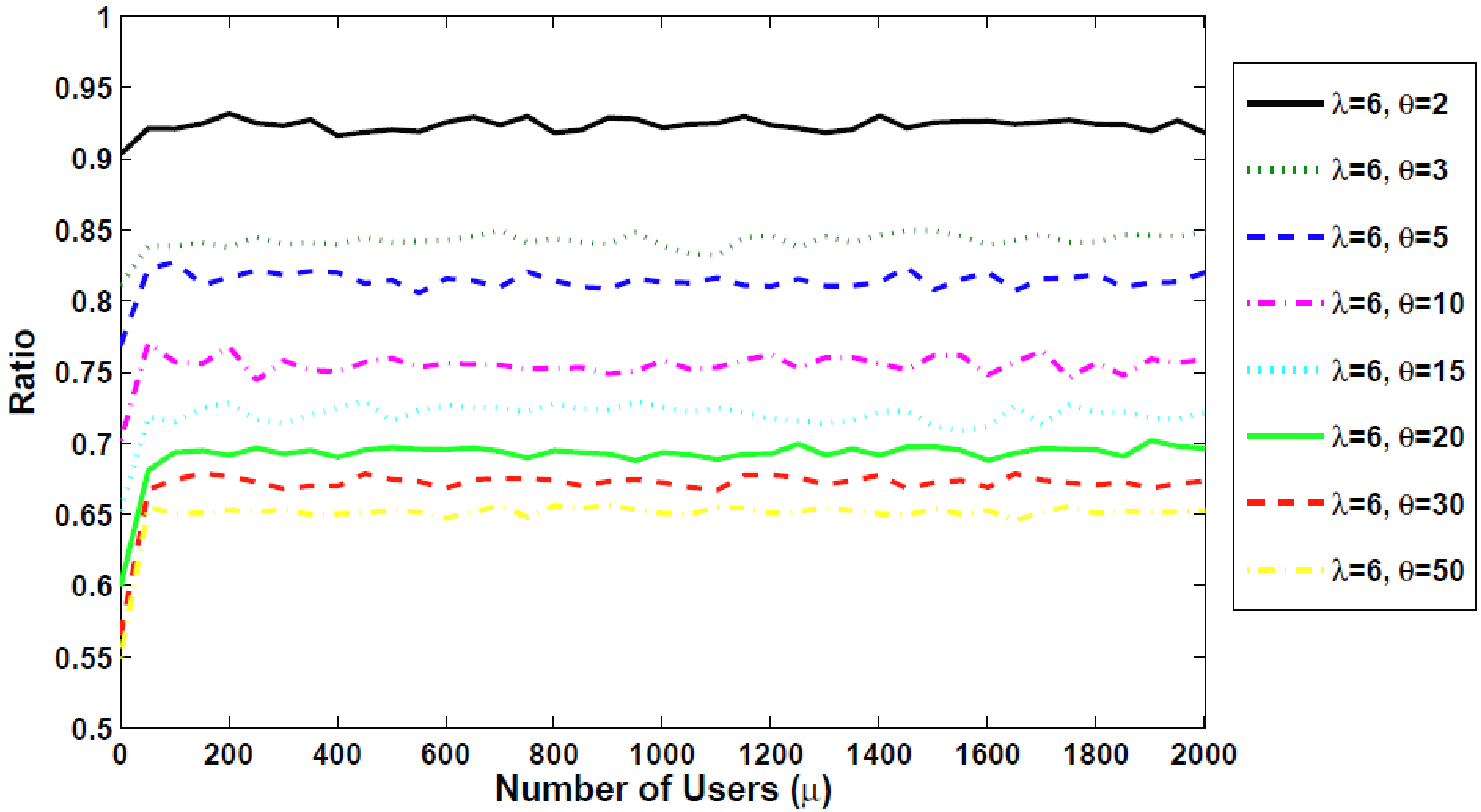
Output: service instance set $\mathcal{S}$, where $\lambda \le |S| \le \gamma, N(F_i) \ge 1$.

```
for u = 1; u ≤ μ; u = u + 1 do
    for k = 1; k ≤ K; k = k + 1 do
        voting(u, k)
for i = 1; i ≤ λ; i = i + 1 do
    I_i ← rank(I_i)
    S = S ∪ I_i.element(0)
I ← rank(I − S)
S = S ∪ I.top(γ − λ)
return S
```

Evaluation – Simple Case

- Evaluation results for simple case

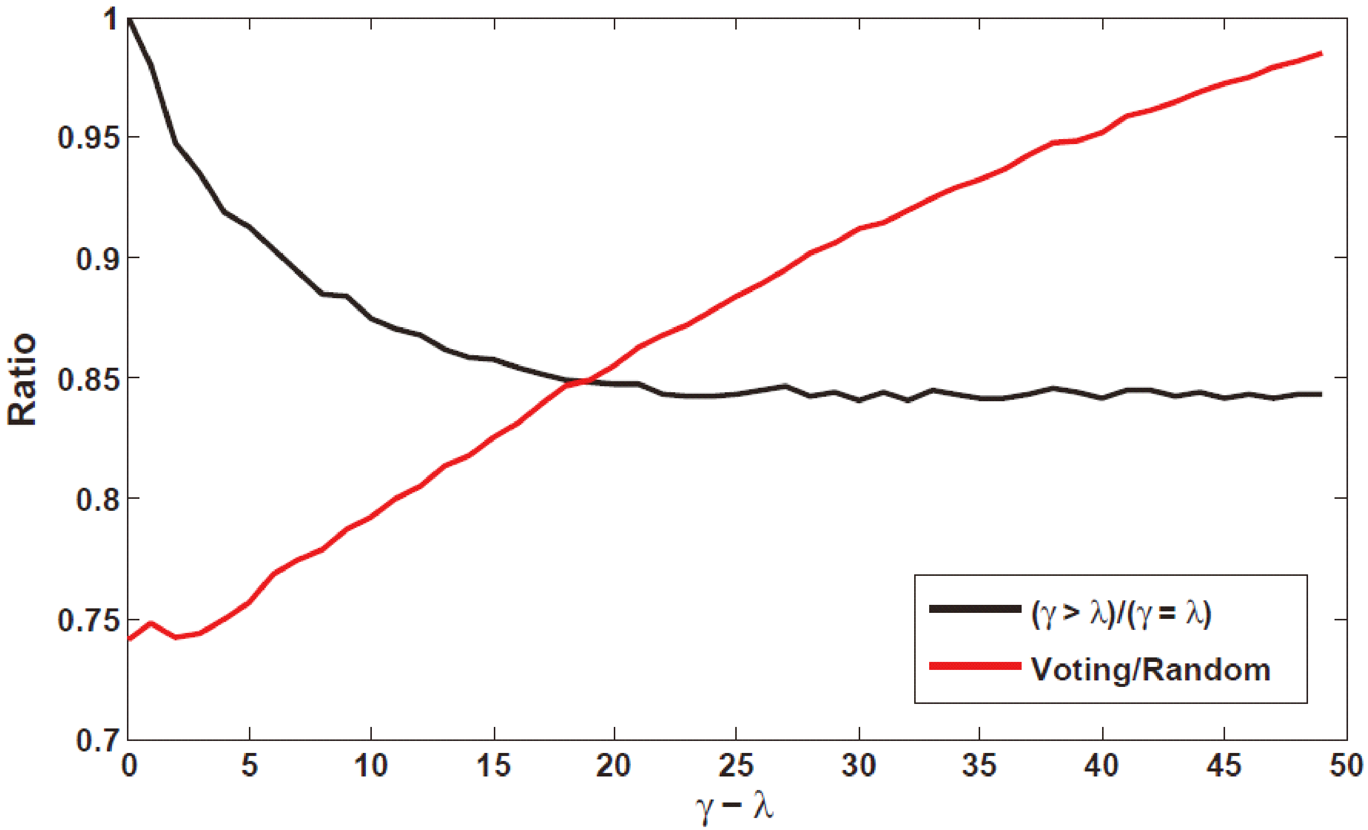
- Impact of



Evaluation – General Case

- Evaluation results for simple case

- Impact of



Thanks!

Q&A