Scaling Service-oriented Applications into Geo-distributed Clouds

香港中文大學
The Chinese University of Hong Kong

Jieming Zhu, Zibin Zheng, Yangfan Zhou, and Michael R. Lyu

Mar. 25, 2013

Presented by

Jieming Zhu
jmzhu@cse.cuhk.edu.hk

Outline

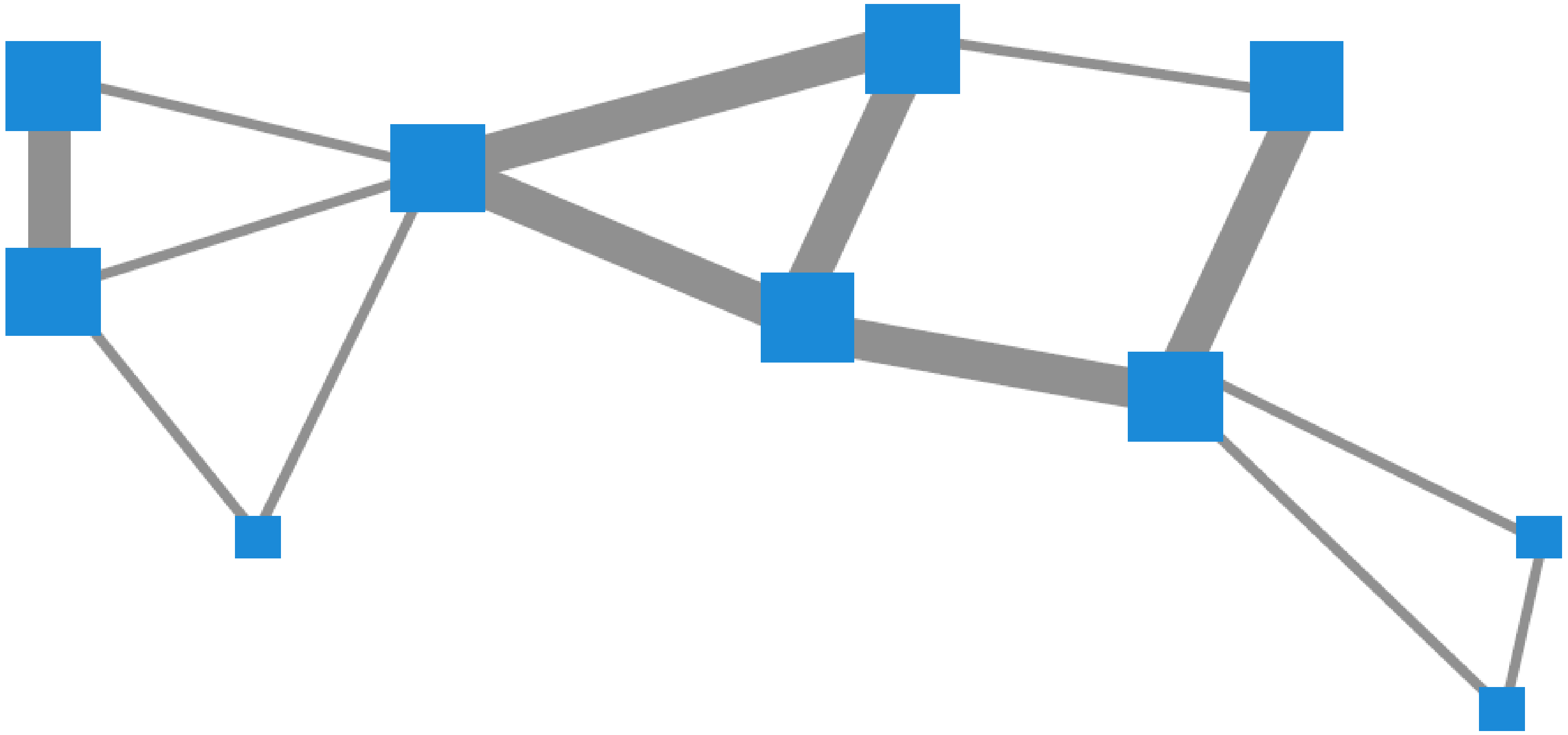
- Introduction
- Challenges
- System Architecture
- Deployment Approach
- Experiments
- Conclusions & Future Work

Introduction

- **Cloud Computing Systems**
  - Scales up
  - Inter-connection

Introduction

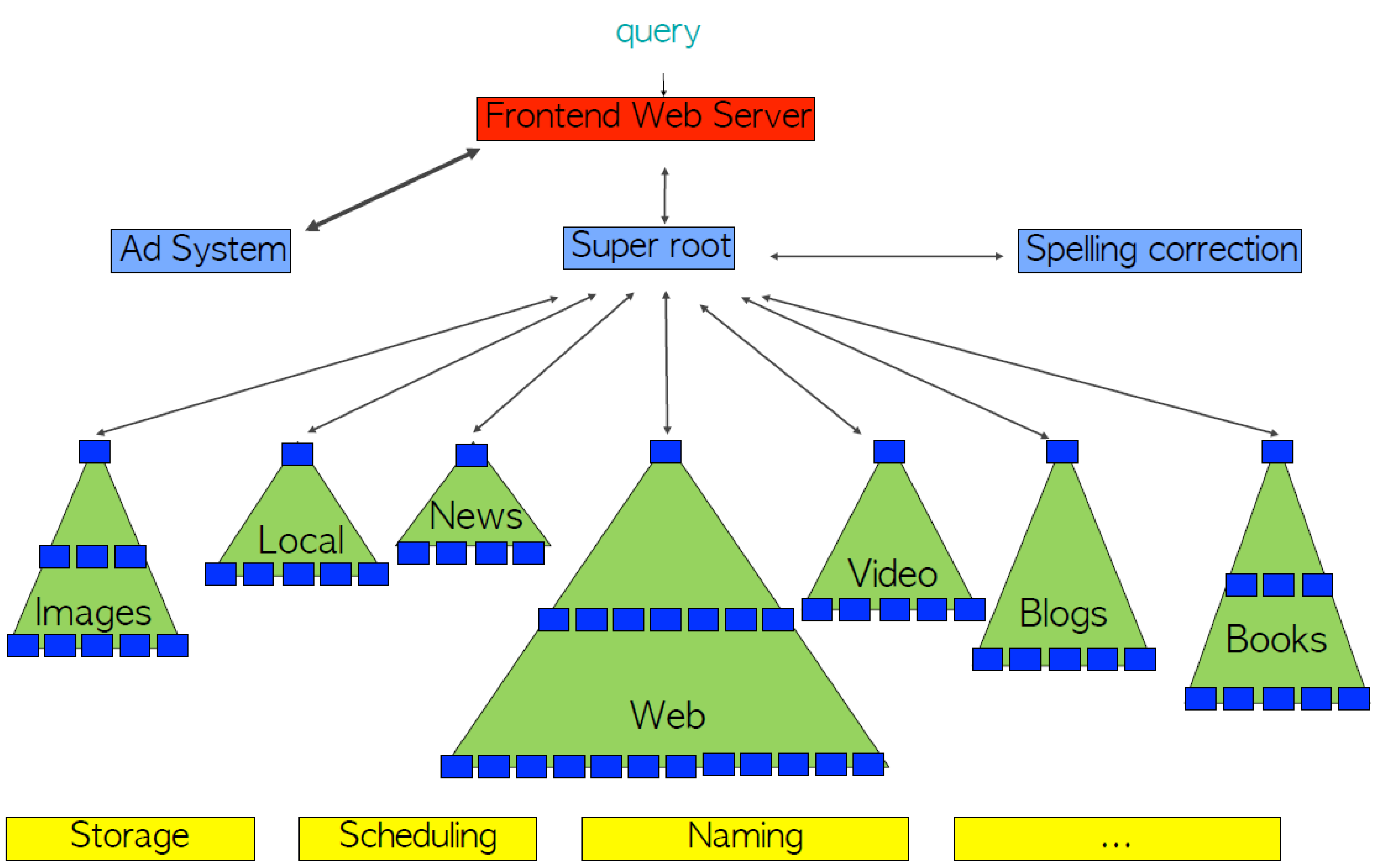
- **Geo-distributed Clouds**
  - Many datacenters around the world
    - E.g., Amazon: Virginia, Oregon, California, Ireland, Singapore, Tokyo, Sydney, S˜ao Paulo, etc.
  - Inter-Cloud / Cloud Federation



[Figure from Google]

Introduction

- **Cloud-based Applications**
  - Large-scale and complex in business logic
  - Service-oriented Architecture
  - Service dependency



[Figure from Google]

Challenges

- **Opportunities:**
  - Take advantage of geo-diversity to
    - Optimize performance: E.g., lower latency
    - Minimize the operational cost

- **Challenges:**
  - Dynamic service demand
  - Dynamic pricing
  - Time-varying communication latencies
  - Geographical distribution of end users
  - Data sharing and interdependency between services in an application

Application Deployement

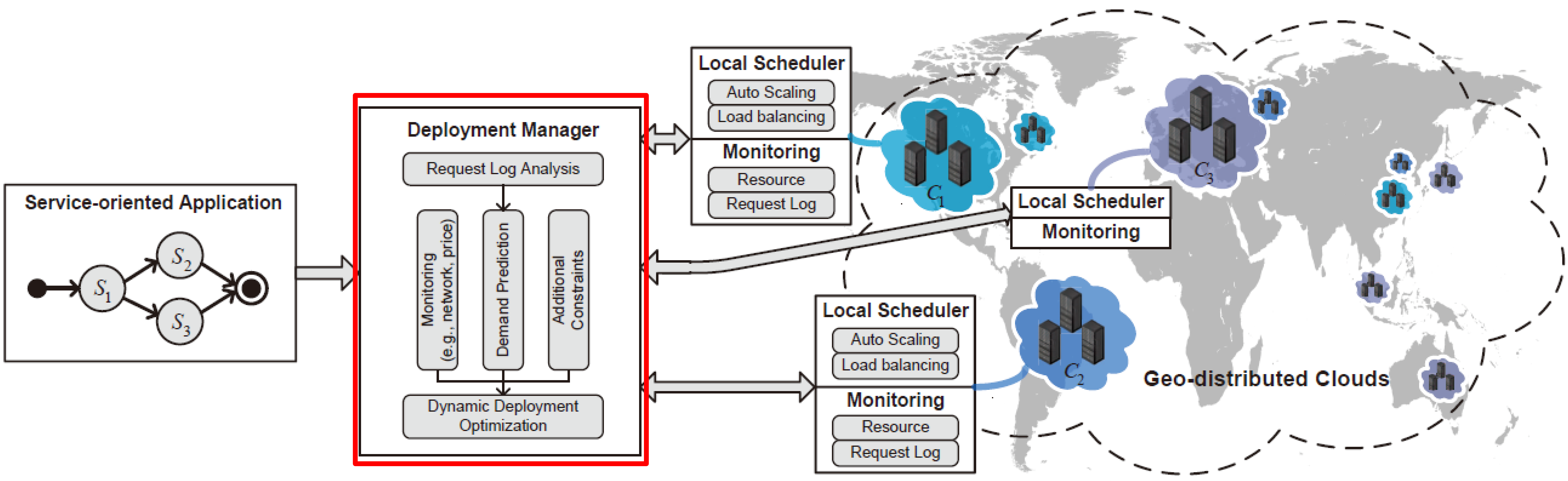
- 1) Which data centers should be selected for each service deployment?

- 2) How many service instances should be replicated for each service in a data center?

System Architecuture

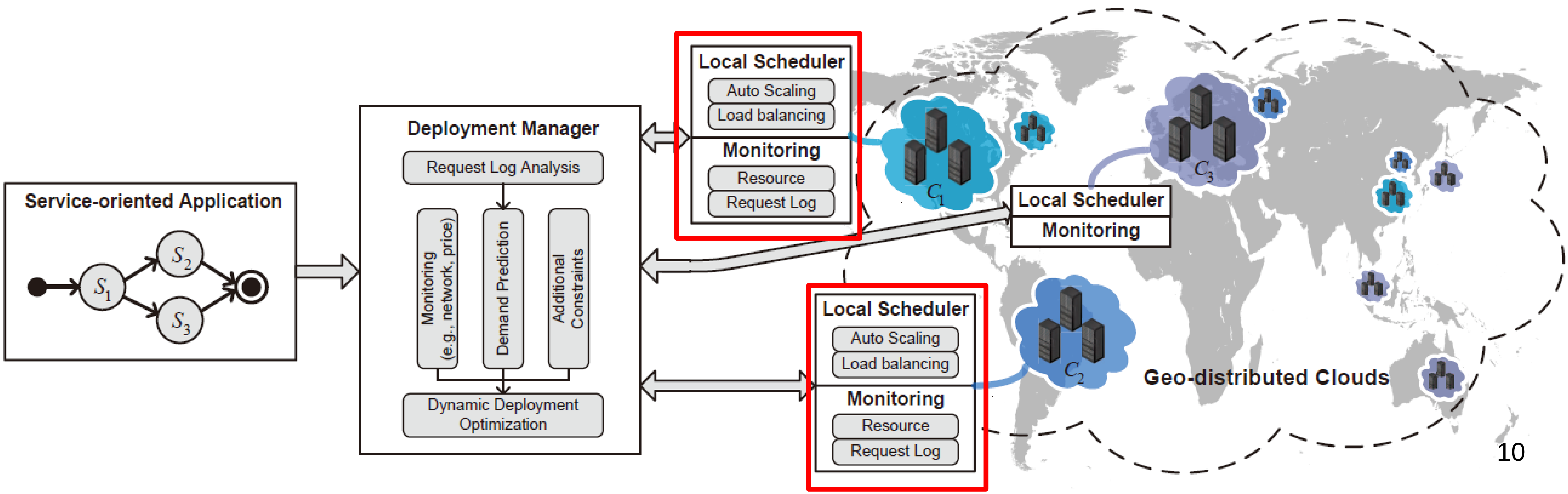
System Architecture

- **Deployment Manager**
  - To determine the locations of each service in the geo-distributed clouds.
    - Monitoring: E.g., network, price
    - Demand prediction
    - Dynamic deployment optimization



System Architecture

- **Local Scheduler**
  - To scale service instances according to the dynamic request workload and route the service requests with load balancing
  - E.g.,
    - **Auto scaling:** http://aws.amazon.com/autoscaling/
    - **Elastic load balancing services:** http://aws.amazon.com/elasticloadbalancing/



Deployment Approach

Deployment Model

- **Objective: Minimize the user-perceived average latency**
- **Constraints:**
  - Operational cost budget $\longrightarrow$ number of service instances K_j
  - Available data center candidates

Model 1 Service Deployment Model across Data Centers

$$\min \sum_{i=1}^{N} \Big(\sum_{j=1}^{M} f_{ij} \cdot \min_{c_j^m \in C_j} d(i, c_j^m) + \sum_{j=1}^{M} \sum_{\substack{k=1 \\ k \neq j}}^{M} f_{jk}^i \cdot \min_{\substack{c_j^m \in C_j \\ c_k^n \in C_k}} d(c_j^m, c_k^n) \Big) \quad (1)$$

$s.t.$

$$C_j \subseteq C, \qquad \forall j = 1, 2, \cdots, M \quad (2)$$

$$|C_j| = K_j, \qquad \forall j = 1, 2, \cdots, M \quad (3)$$

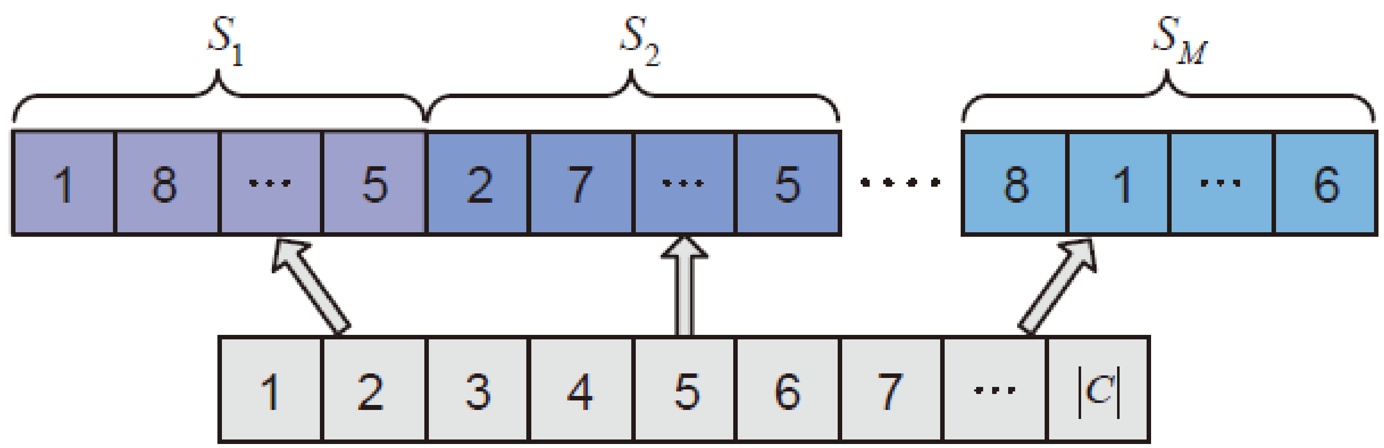
TABLE I. NOTATIONS OF DEPLOYMENT MODEL

Notations	Descriptions
N	Number of users
M	Number of service components in an application
f_{ij}	Frequency of invocations between user i and service j
c_j^m	The m-th datacenter deployed by service j
$d(i, c_j^m)$	Latency between user i and datacenter c_j^m
f_{jk}^i	Frequency of invocations between service j and service k for user i
C_j	Deployment strategy of servie j (the seleted data centers for service j)
C	The set of candidate datacenters
K_j	The number of instances of service j

TABLE I. NOTATIONS OF DEPLOYMENT MODEL

Genetic Algorithm Design

- Gene Representation



Candidate Data Center Set

- Fitness function
- Parents selection
- Crossover
- Mutation

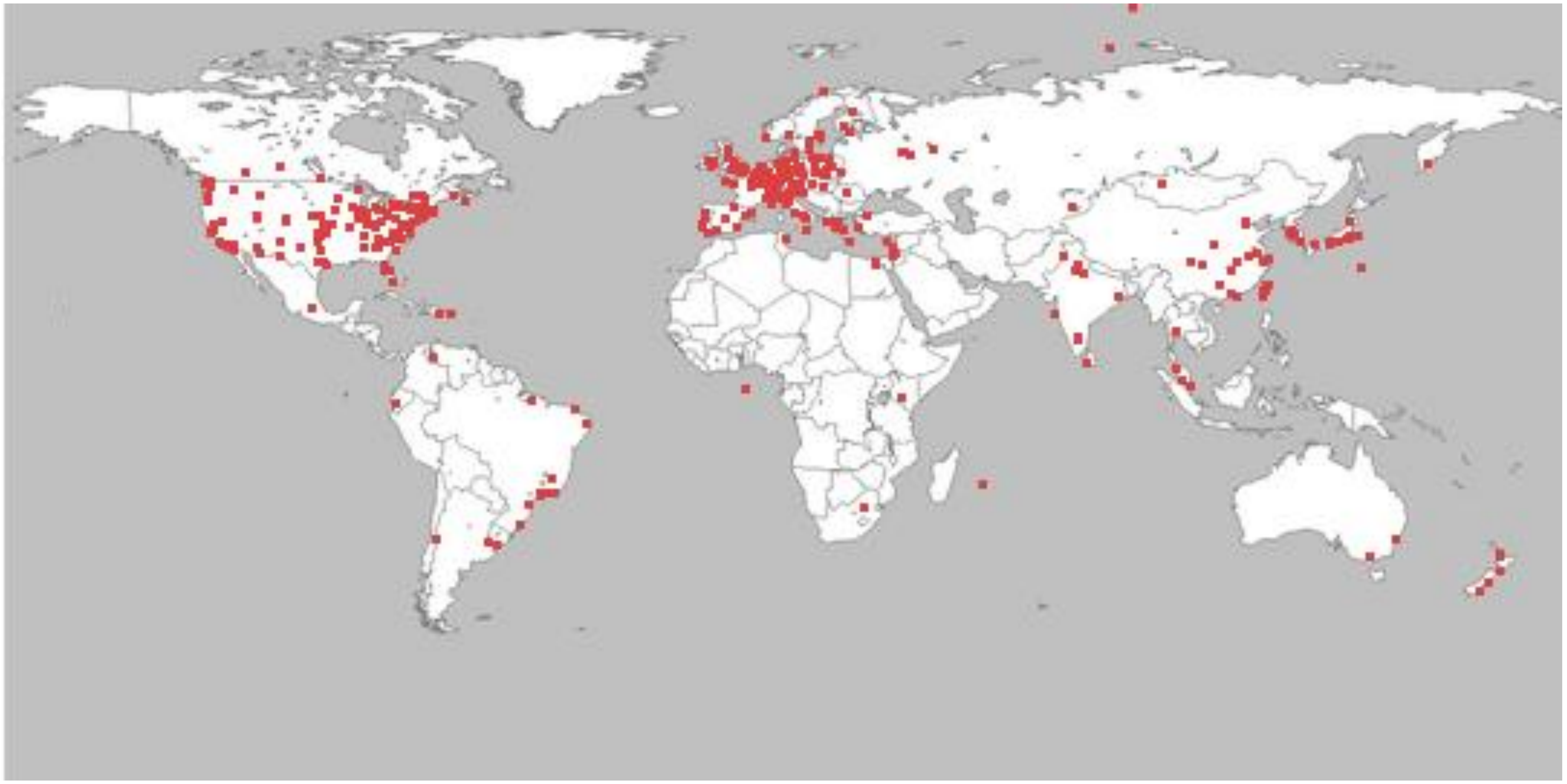
Algorithm 1: Genetic Algorithm based Deployment Algorithm

```
Input: Fitness function, VMs candidates
Output: The best objective value Latency, the deployment solution C_k
Generate initial population;
Compute the fitness value;
gen = 0;
while gen < MAXGEN do
    Fitness scaling;
    Select the parents;
    Crossover;
    Mutation;
    Compute the fitness value of offsprings;
    Reinsertion;
    gen++;
end
return Latency, C_k
```

Experiments

Experiments

- **Dataset description**
  - Planetlab nodes distribution



  - Dataset:
    - User-DC matrix: 1881×307
    - Inter-DC matrix: 307×307

Experiments

- **Dataset description**
  - Latency distribution

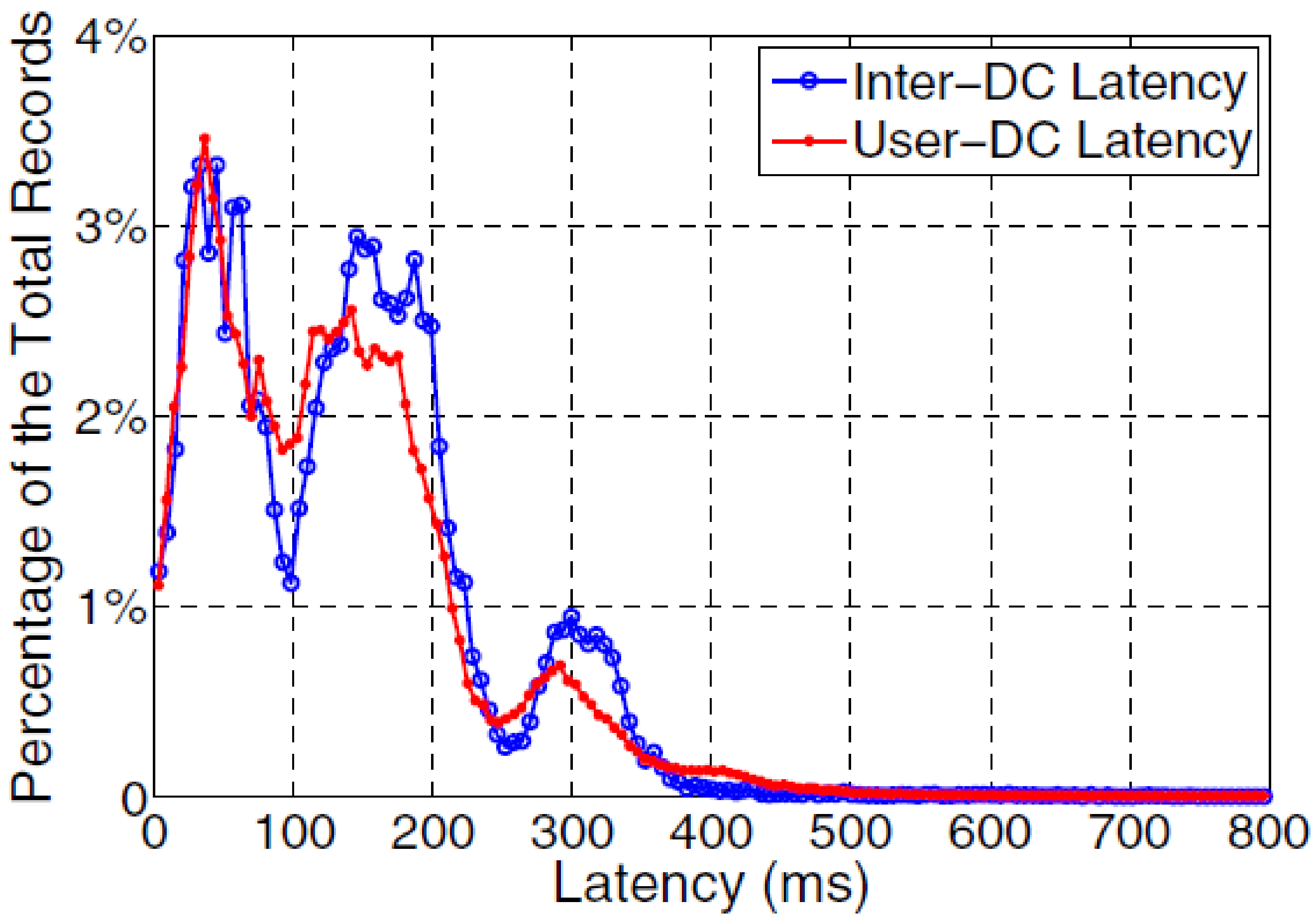
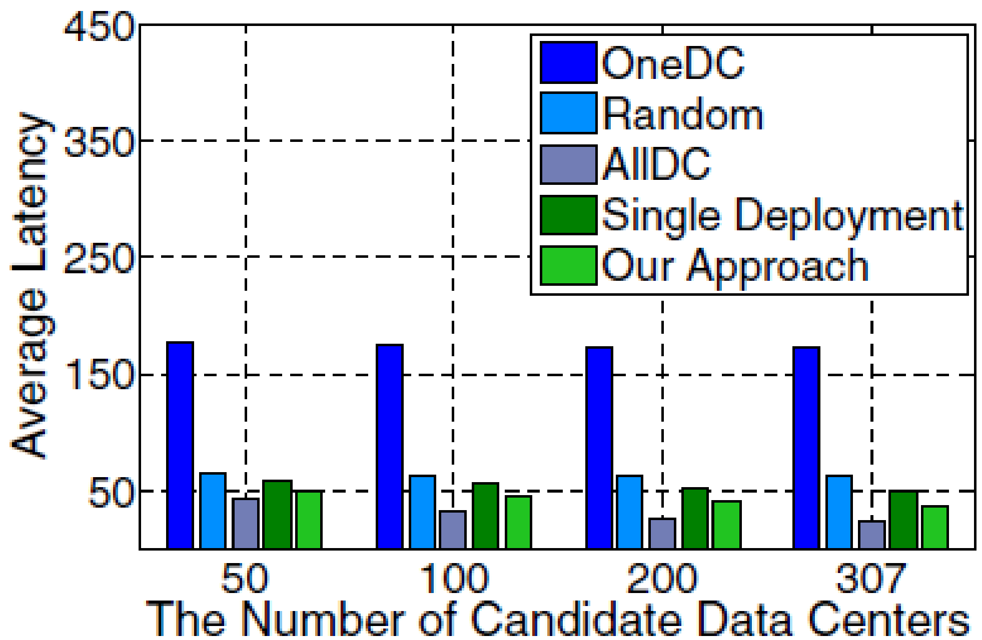


Fig. 3. The Approximate Distributions of Inter-DC Latencies and User-DC Latencies (DC: Data Center)

Experiments

- **Performance Comparison**
  - **Random:** The services are deployed randomly in K_j data centers.
  - **OneDC:** Deploys all the services in one data center with the highest performance of all the candidates.
  - **AllDC:** Simply deploys each service in every data center.
  - **Single Deployment:** Optimizes the deployment strategy independently for each single service



(a) Performance Comparison

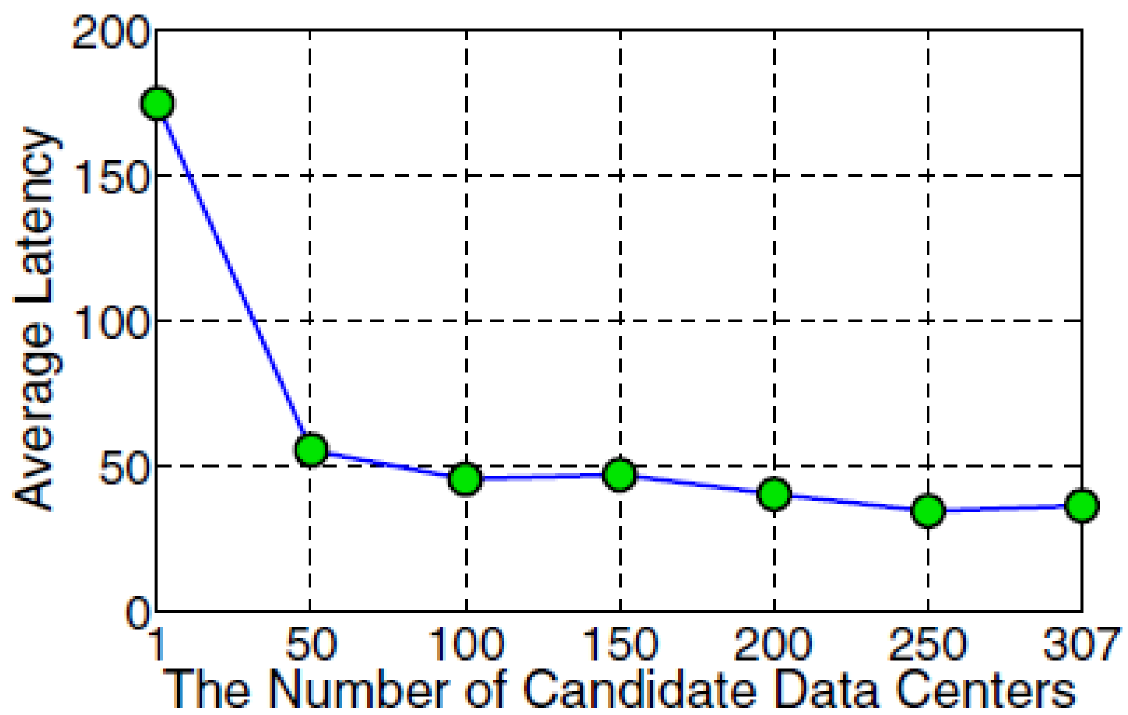
User logs: randomly generated

Parameters:

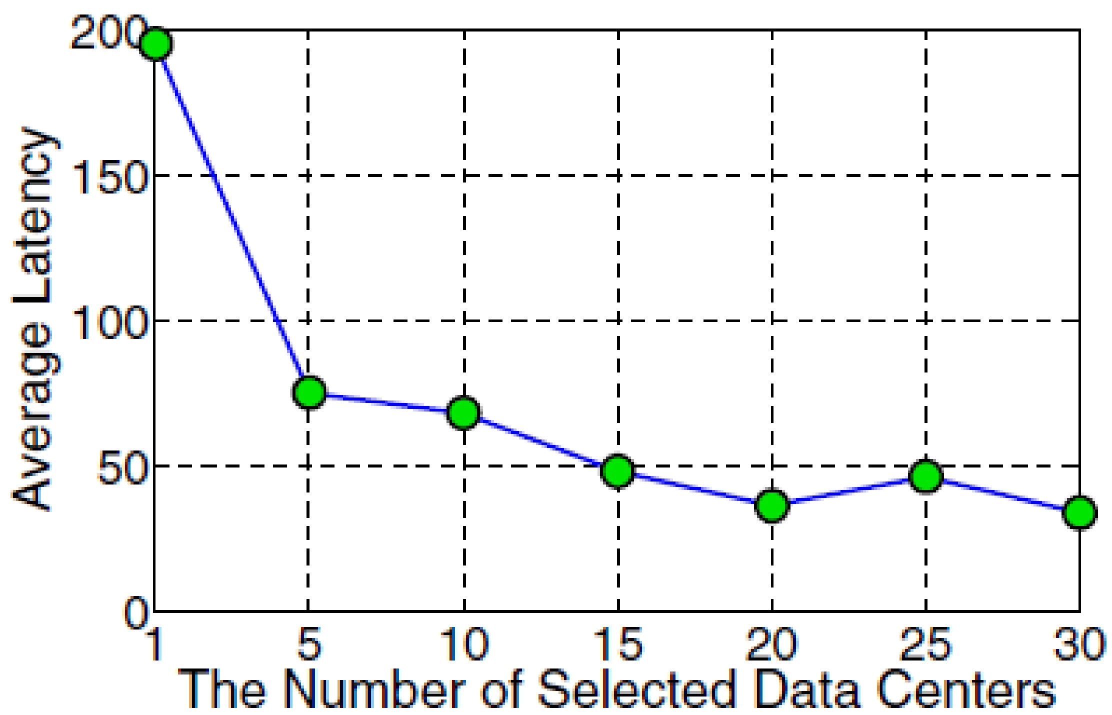
$N = 1881$, $M = 10$, and $|C| = 100$

Experiments

- Impact of parameters



(b) Impact of $|C|$



(c) Impact of K_j

- Find more in our paper

Conclusion

- **Contributions:**
  - We propose a general framework to address the problem of dynamic service-oriented application deployment in geo-distributed clouds
    - Optimizing the application performance
    - Keeping minimal operational cost.
  - Exploits service dependencies to optimize the deployment strategy across a set of candidate data centers
  - Extensive experiments are conducted based on a large-scale real-world latency dataset
    - Dataset download: http://www.wsdream.net
- **Future work:**
  - Explore how to improve our deployment model by incorporating the capacity and cost models
  - Load balancing strategies

Thank you!

Q & A