香港中文大學
The Chinese University of Hong Kong

Data-Driven Quality Management of Online Service Systems

ZHU, Jieming
Supervisor: Prof. Michael R. Lyu

2015/12/14

Online services are serving many aspects of our daily life

Popular online services

Web search

Social network

Online chatting

Online shopping

And many others…

7 x 24 availability

Responsiveness

....

Quality of service

is critical to success

User satisfaction

Revenue increase

Quality degradation causes revenue loss

Google's 5-minute outage means $545,000 revenue loss, 40% drop in global website traffic

By George Tinari · Aug 18, 2013 · HOT!

90

Amazon just lost $4.8M after going down for 40 minutes

BY TAYLOR SOPER on August 19, 2013 at 1:01 pm

9 Comments | Share 88 | Tweet | Share 39 | Reddit | Pin | Join us this Thursday for GeekWire Radio

When you're an online retail giant, every second that your website stays inaccessible means thousands of dollars out the virtual door.

For Amazon, that meant about $5 million in losses Monday afternoon when the Seattle company went down for about 40 minutes.

Amazon posted $15.7 billion in revenue last quarter, which breaks down to about $120,000 per minute. Multiply that by the 40 minutes Amazon went down and you get around $4.8 million in lost dollars.

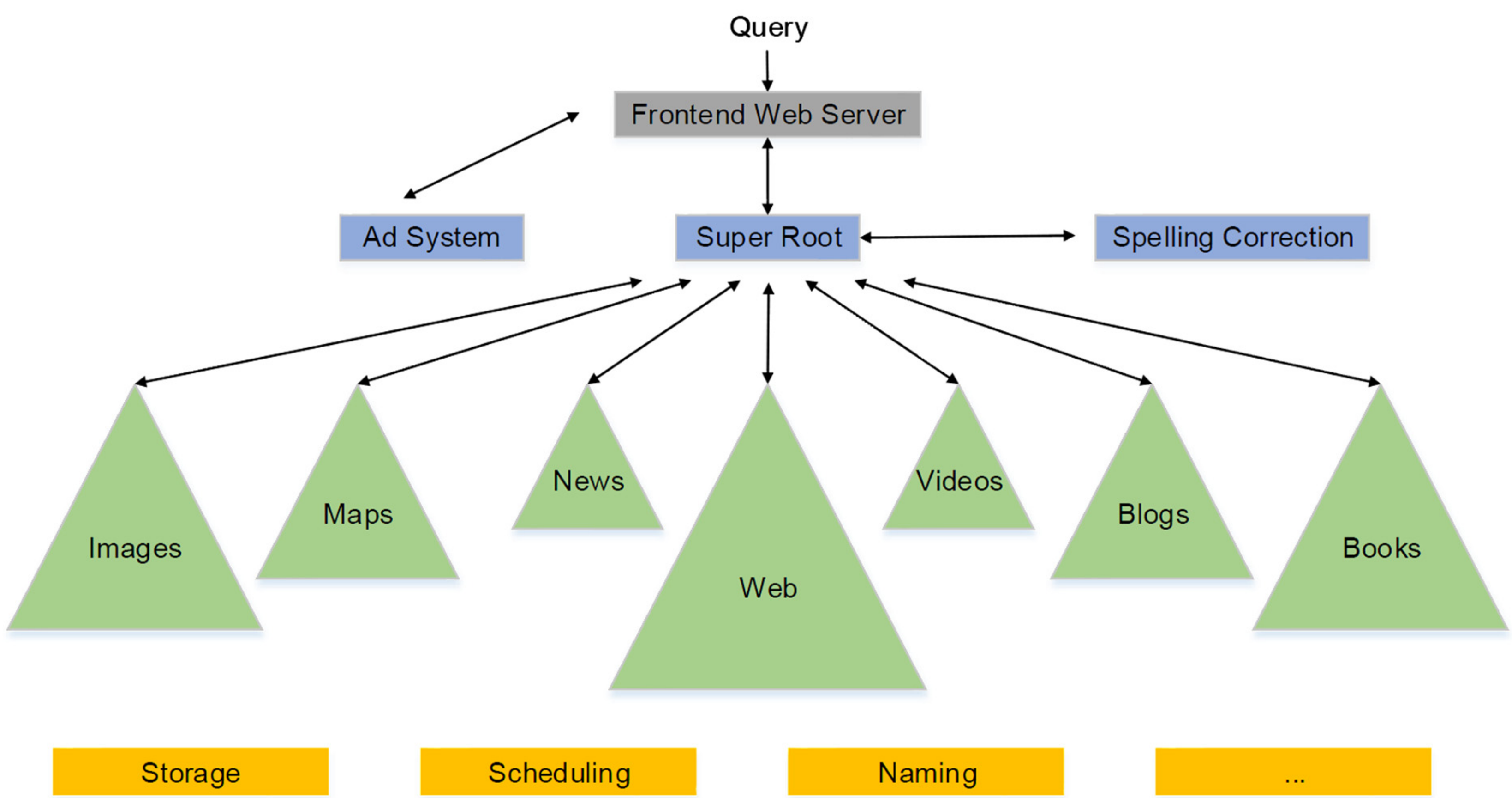
amazon web services™

Quality management of online service systems **is important**, **but challenging**

Online service systems are becoming **large-scale in size** and **complex in structure**

[Image from: http://www.lifehack.org/articles/technology/you-may-never-know-these-google-search-tips-and-tricks-you-miss-this.html]

Online service systems are built on **service-oriented architectures**

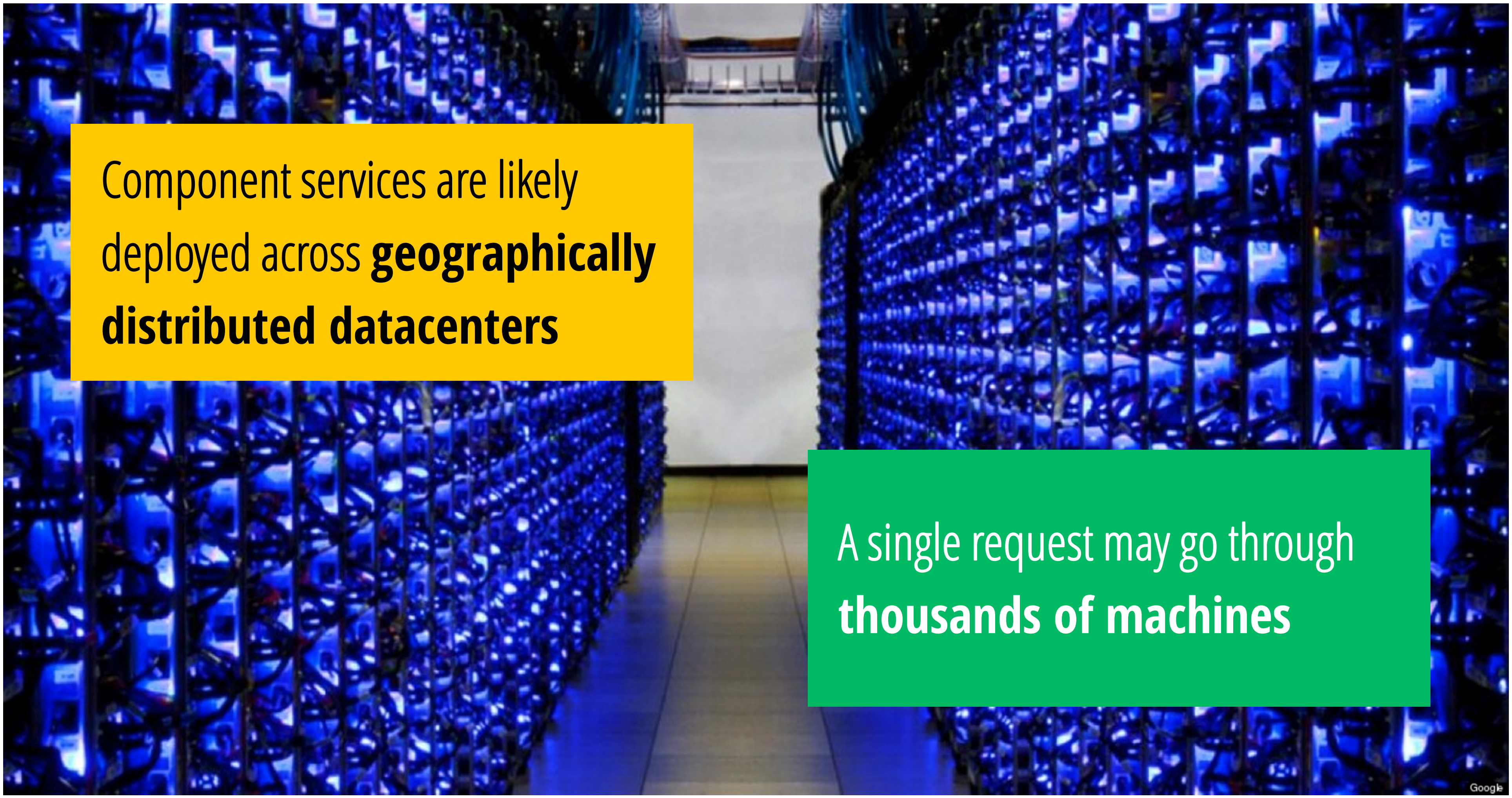


A prototype of Google's system

[Image adapted based on Jeff Dean's slides: http://www.slideshare.net/yarapavan/achieving-rapid-response-times-in-large-online-services]

CHALLENGE

Online service systems are **highly distributed**



Component services are likely deployed across **geographically distributed datacenters**

A single request may go through **thousands of machines**

[Image from: http://www.slideshare.net/yarapavan/achieving-rapid-response-times-in-large-online-services]

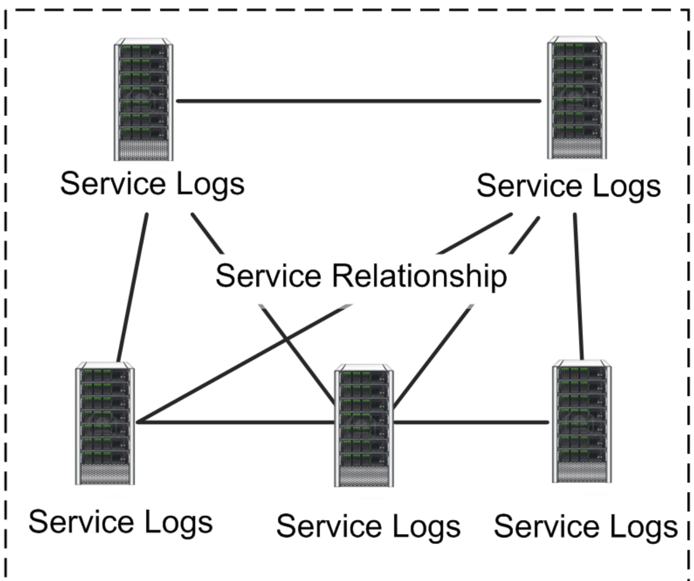
Traditional engineering techniques
are often not sufficient

Data-driven service quality management is in need

Data-driven service quality management

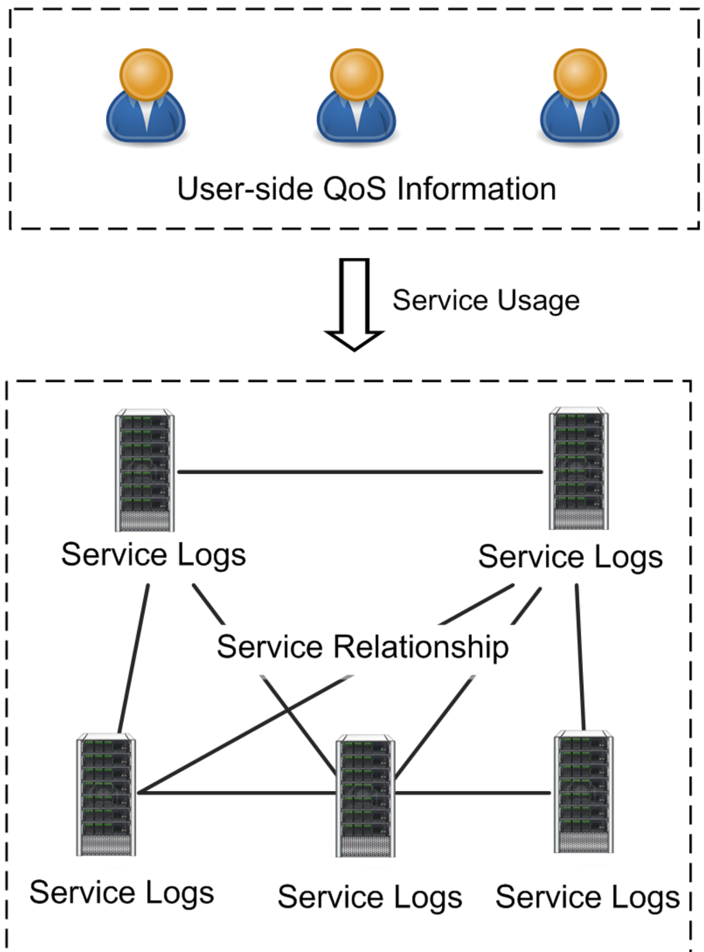
Service-generated logs

Data-driven service quality management



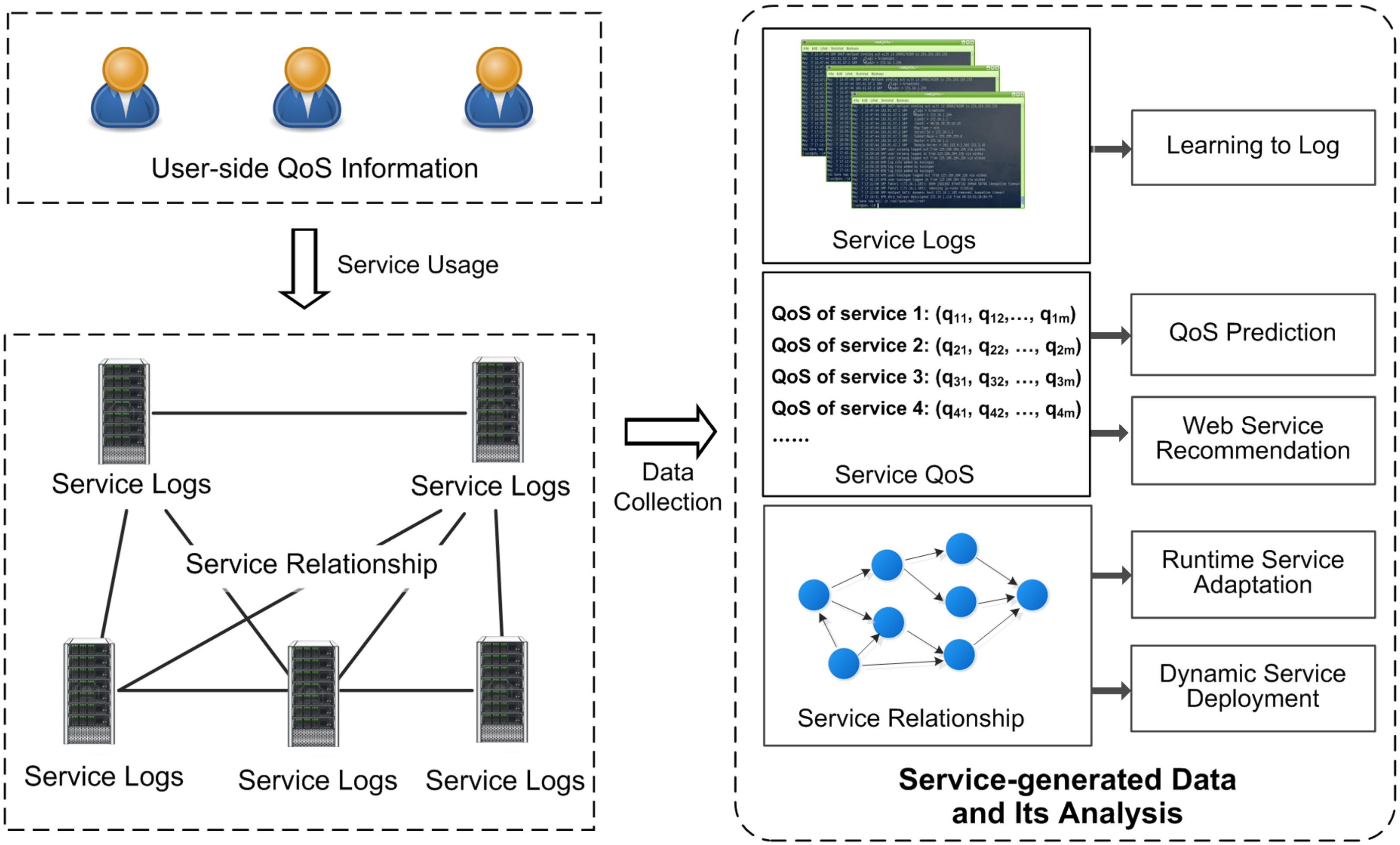
Service relationship information

Data-driven service quality management

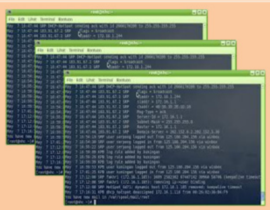


User-perceived QoS (Quality of Service) information

Data-driven service quality management



Thesis contributions



Service Logs

→ Learning to log for runtime service monitoring [ICSE'15, ICSE'14] (Chapter 6)

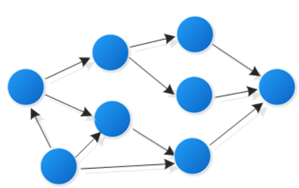
QoS of service 1: $(q_{11}, q_{12}, \ldots, q_{1m})$
QoS of service 2: $(q_{21}, q_{22}, \ldots, q_{2m})$
QoS of service 3: $(q_{31}, q_{32}, \ldots, q_{3m})$
QoS of service 4: $(q_{41}, q_{42}, \ldots, q_{4m})$
……

Service QoS

→ Response time prediction [ICWS'12, iVCE'12] (Chapter 3)

→ Online QoS prediction [ICDCS'14] (Chapter 4)

→ Privacy-preserving QoS prediction [ICWS'15] (Chapter 5)



Service Relationship

→ Dynamic service deployment [iVCE'13] (Chapter 3.5)

→ Dynamic request routing [CLOUD'13] (Chapter 4.5)

Outline

- **Topic 1: Learning to log** for runtime service monitoring
- **Topic 2: Online QoS prediction** of Web services
- Conclusion and future work

Outline

- **Topic 1: Learning to log** for runtime service monitoring
- **Topic 2: Online QoS prediction** of Web services
- Conclusion and future work

Outline

- **Topic 1: Learning to log** for runtime service monitoring
  - Motivation
  - Framework of learning to Log
  - Implementation details
  - Evaluation
  - Summary

What is logging?

Logging is a **common programming practice** to record runtime system information

Logging format: *Log (level, "logging message %s", variable)*;

Log example:

```
Failed password for root from 10.0.0.132 port 57807 ssh2
```

Logging methods

- **Basic utilities**: printf, cout, writeline
- **Sophisticated tools**: log4j, Unified Logging System (Microsoft)

The importance of logging

Logs are used as **a principal tool** for runtime service monitoring

- Usage analysis
- Anomaly detection
- Failure diagnosis
  - The only data available for diagnosing production failures

Commercial acceptance

- Vendors actively collect logs: Microsoft, Google, VMware

Logging is significantly important!

Challenges of logging

Logging too little

– Miss valuable runtime information
– Increase the difficulty for problem diagnosis

User:
"Apache httpd cannot start.
No log message printed."

[Yuan et al., OSDI'12]

Logging too much

– Additional cost of code development & maintenance
– Runtime overhead (CPU, I/O)
– Too much redundant/useless logs

Challenges of logging

Logging too little

- Miss valuable runtime information
- Increase the difficulty for problem diagnosis

"Apache httpd cannot start.

[Yuan et al., OSDI'12]

Logging too much

- Additional cost of code development & maintenance
- Runtime overhead (CPU, I/O)
- Too much redundant/useless logs

Developers need to make informed logging decisions on where to log!

Current practice of logging

An empirical study on logging practice [ICSE'14]

- Developer survey
  - 37 developers participated (~4.9 years of programming experience)
- Source code analysis
  - 4 large software systems from both Microsoft and Github

How do developers make logging decisions in industry?

- Lack of rigorous specifications on logging
- Mostly based on domain knowledge of developers

Contributions of this work

Learning to log for runtime service monitoring

– Automatically learn logging practice from existing logging instances via machine learning

– Provide logging suggestions during development

– Implemented as a prototype tool "LogAdvisor"

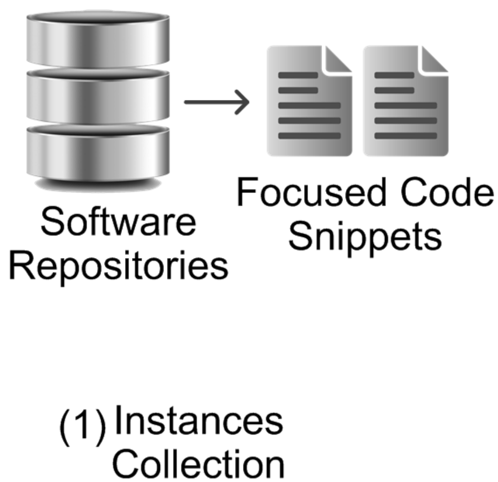
The work was collaborated with Microsoft Research Asia

Outline

- **Topic 1: Learning to log** for runtime service monitoring
  - Motivation
  - Framework of learning to Log
  - Implementation details
  - Evaluation
  - Summary

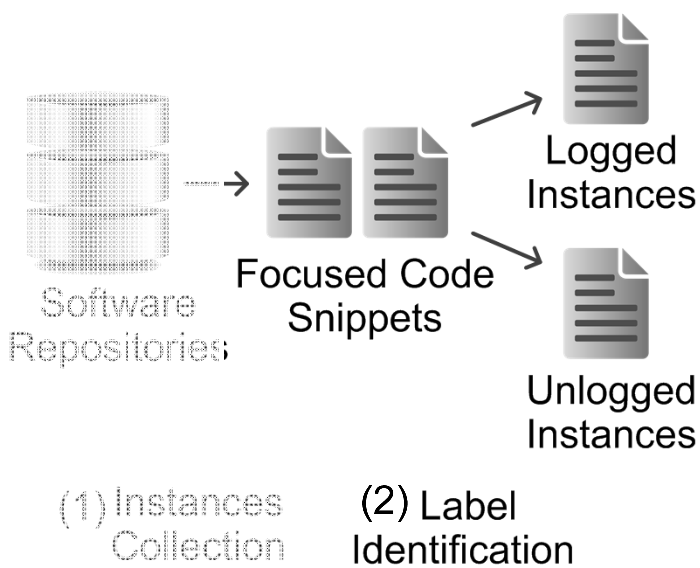
Framework of learning to log

A general learning framework similar to other machine learning applications



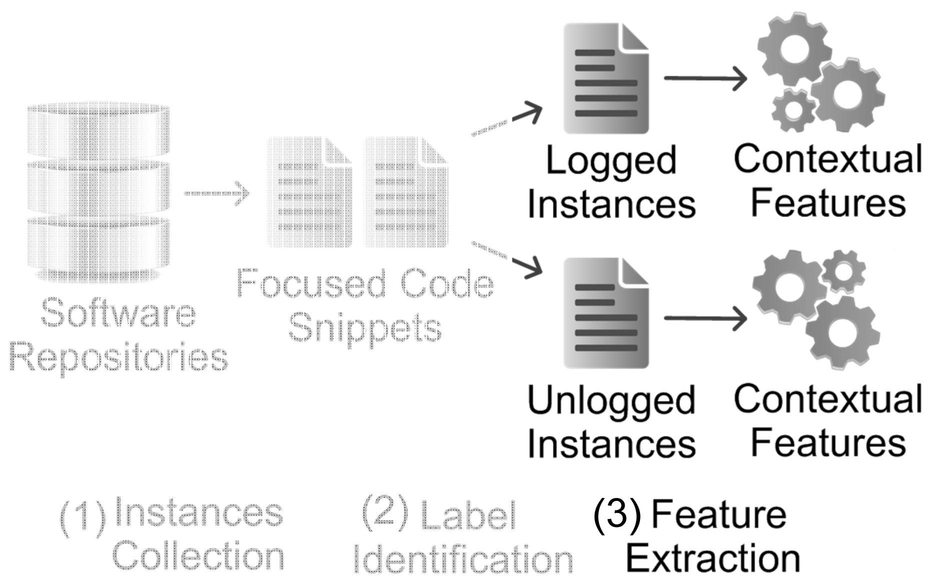
Framework of learning to log

A general learning framework similar to other machine learning applications



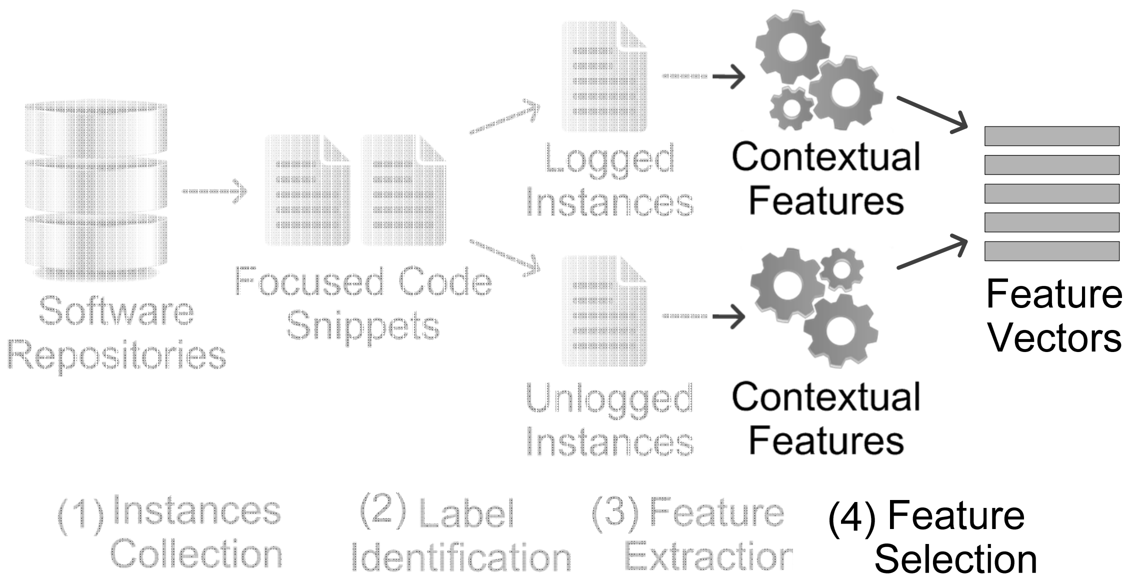
Framework of learning to log

A general learning framework similar to other machine learning applications



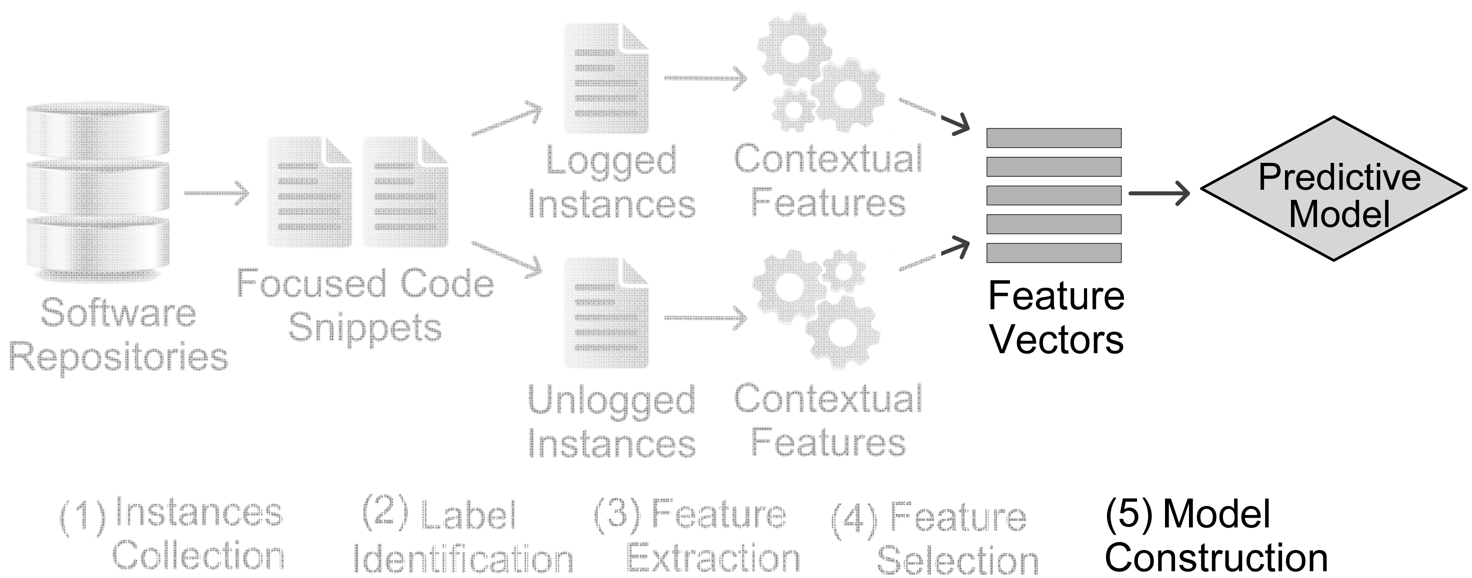
Framework of learning to log

A general learning framework similar to other machine learning applications



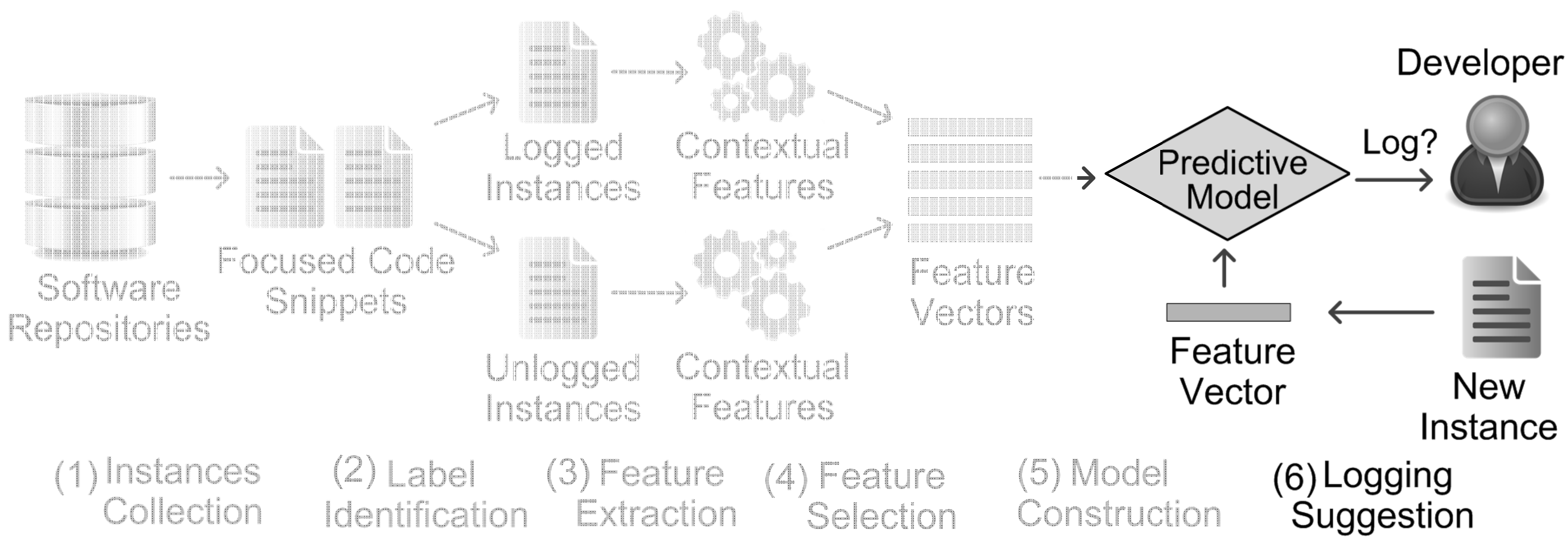
Framework of learning to log

A general learning framework similar to other machine learning applications



Framework of learning to log

A general learning framework similar to other machine learning applications



Outline

- **Topic 1: Learning to log** for runtime service monitoring
  - Motivation
  - Framework of learning to Log
  - Implementation details
  - Evaluation
  - Summary

(1) Instances Collection	(2) Label Identification	(3) Feature Extraction	(4) Feature Selection	(5) Model Construction	(6) Logging Suggestion

Focused snippets: indicate potential error sites

- Exception snippets: try-catch blocks
- Return-value-check snippets: function-return errors

Exception snippet example

```
try {
    method(…);
}
catch (IOException) {
    log(…);
    …
}
```

Return-value-check snippet example

```
var res = method(…);
if (res == null) {
    log(…);
    …
}
```

(1) Instances Collection | (2) Label Identification | (3) Feature Extraction | (4) Feature Selection | (5) Model Construction | (6) Logging Suggestion

All the code analysis is conducted based on an open-source C# code analysis tool, **Roslyn**

Label identification

– "logged" if a focused code snippet contains a logging statement
– "unlogged", otherwise.

```
try {
        method(…);
}
catch (IOException) {
        log(…);
        …
}
```

logged

```
var res = method(…);
if (res == null) {
        log(…);
        …
}
```

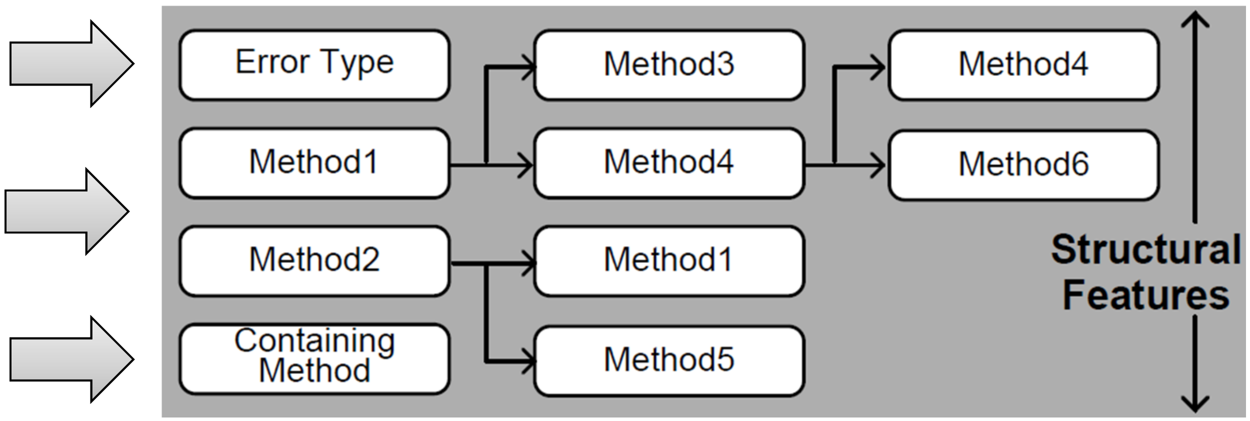
logged

Contextual feature extraction

– Structural features
– Textual features
– Syntactic features

Feature extraction (1)

Structural features: structural info of code




```
private int LoadRulesFromAssembly (string assembly, …){
        //Code in Setting
        try {
                AssemblyName aname = AssemblyName.
                GetAssemblyName(Path.GetFullPath (assembly));
                Assembly a = Assembly.Load (aname);
        }
        catch (FileNotFoundException) {
                Console.Error.WriteLine ("Could not load rules
                From assembly '{0}'.", assembly); return 0; }
        … }
}
```

Exception Type:
System.IO.FileNotFoundException

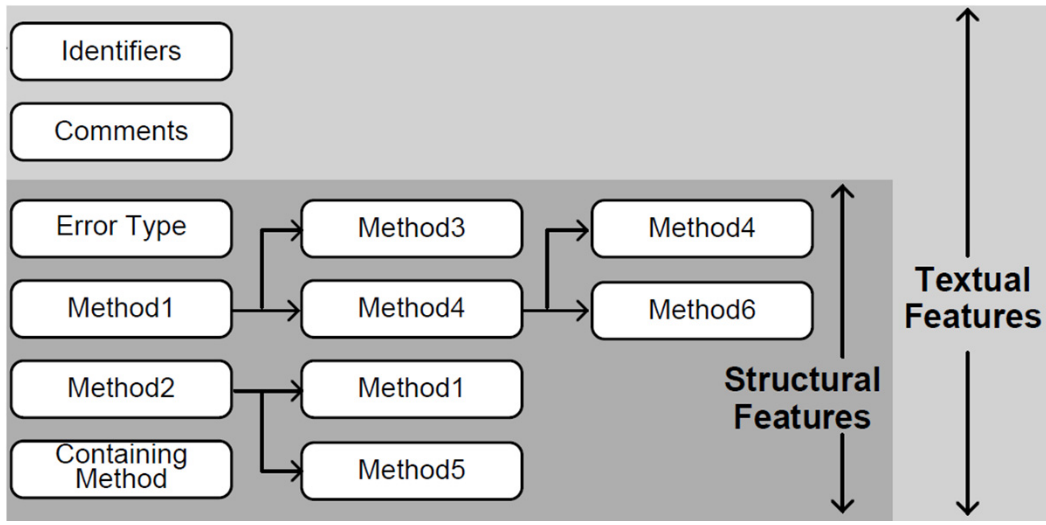
Containing method:
Gendarme.Settings.LoadRulesFromAssembly

Invoked methods:
System.IO.Path.GetFullPath,
System.Reflection.AssemblyName.GetAssemblyName,
System.Reflection.Assembly.Load

/* A code example taken from MonoDevelop (v.4.3.3), at file: * main\external\mono-tools\gendarme\console\Settings.cs,
* line: 116. Some lines are omitted for ease of presentation. */

Feature extraction (2)

Textual features: code as text



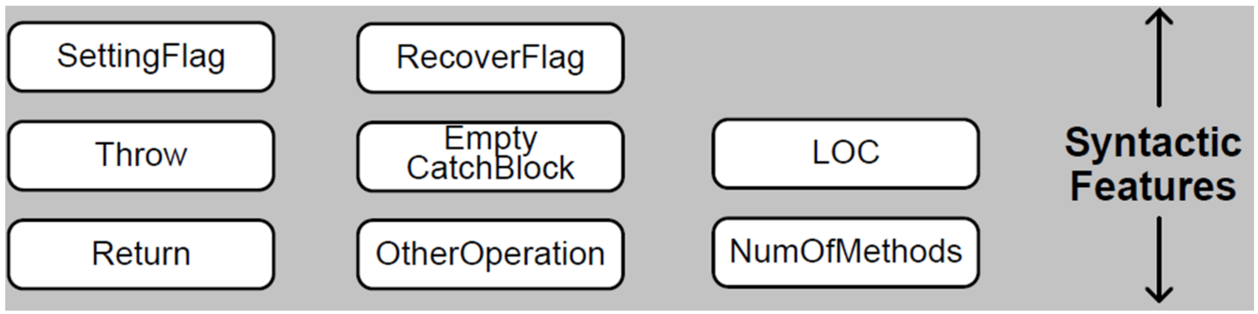

```
private int LoadRulesFromAssembly (string assembly, ...){
        //Code in Setting
        try {
                AssemblyName aname = AssemblyName.
                GetAssemblyName(Path.GetFullPath (assembly));
                Assembly a = Assembly.Load (aname);
        }
        catch (FileNotFoundException) {
                Console.Error.WriteLine ("Could not load rules
                From assembly '{0}'.", assembly); return 0; }
        ... }
}
```

Textual features:

load(2), rules(1), assembly(7), setting(1), name(2), aname(2), get(2), path(1), full(1), file(1), not(1), found(1), exception(1)

Feature extraction (3)

Syntactic features: syntactic info of code




```
private int LoadRulesFromAssembly (string assembly, ...){
        //Code in Setting
        try {
                AssemblyName aname = AssemblyName.
                GetAssemblyName(Path.GetFullPath (assembly));
                Assembly a = Assembly.Load (aname);
        }
        catch (FileNotFoundException) {
                Console.Error.WriteLine ("Could not load rules
                From assembly '{0}'.", assembly); return 0; }
        ... }
}
```

```
Syntactic features:
SetLogicFlag: 0          EmptyCatchBlock: 0
Throw: 0                 OtherOperation: 0
Return: 1                LOC: 3
RecoverFlag: 0           NumOfMethods: 3
```

Feature selection

High-dimensional feature vectors (~72K features in System-B)

- Remove infrequence features (e.g., less than 5)
- Leverage information gain for further elimination

Data imbalance handling

- Unlogged vs logged instances (ratio up to 50 : 1)
- Unlogged instances dominate the neighborhood
- Use **SMOTE** [Chawla et al., 2002] to balance data

(1) Instances Collection	(2) Label Identification	(3) Feature Extraction	(4) Feature Selection	(5) Model Construction	(6) Logging Suggestion

- **Classification models**
  - Naive Bayes
  - Bayes Net
  - Logistic Regression
  - SVM
  - Decision Tree
- Providing **logging suggestions** by using constructed models: whether or not to log a code snippet

Outline

- **Topic 1: Learning to log** for runtime service monitoring
  - Motivation
  - Framework of learning to Log
  - Implementation details
  - Evaluation
  - Summary

Systems under study

Four large-scale software systems

- **System-A** and **System-B** (anonymized)
  - Production online service systems from Microsoft
- **SharpDevelop** and **MonoDevelop**
  - Open-source projects from Github
  - Popular C# projects
  - 10000+ commits
  - 10+ years of history

C# software systems, 19.1M LOC, 100.6K logging instances in total

Evaluation setup

Ground truth: logging labels made by code owners

Metric: balanced accuracy (BA)

$$BA = \frac{1}{2} \times \underbrace{\frac{TP}{TP+FN}}_{\text{Accuracy of logged instances}} + \frac{1}{2} \times \underbrace{\frac{TN}{TN+FP}}_{\text{Accuracy of unlogged instances}}$$

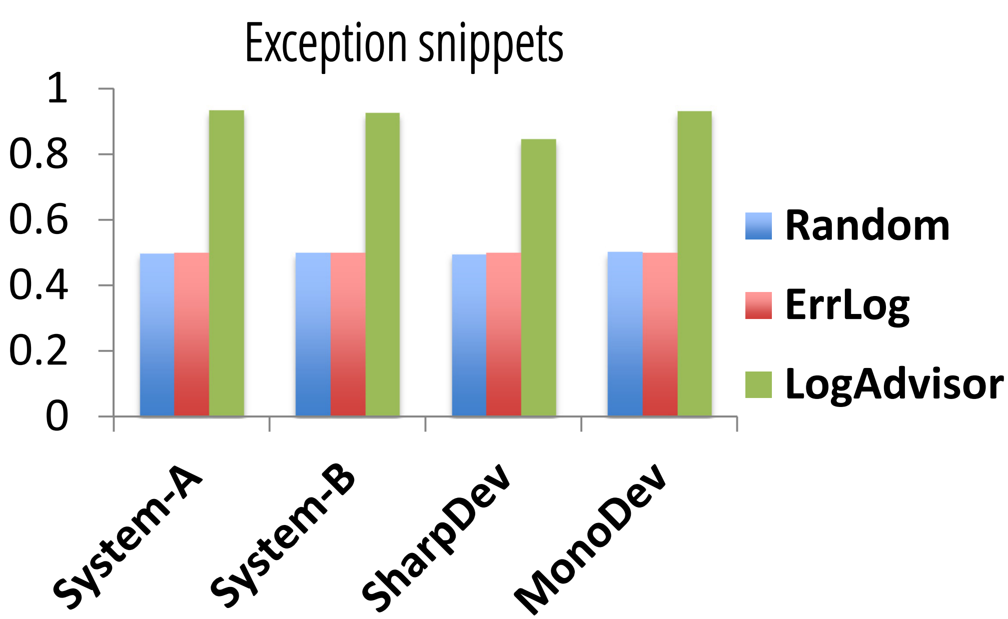
Within-project evaluation: 10-fold cross evaluation

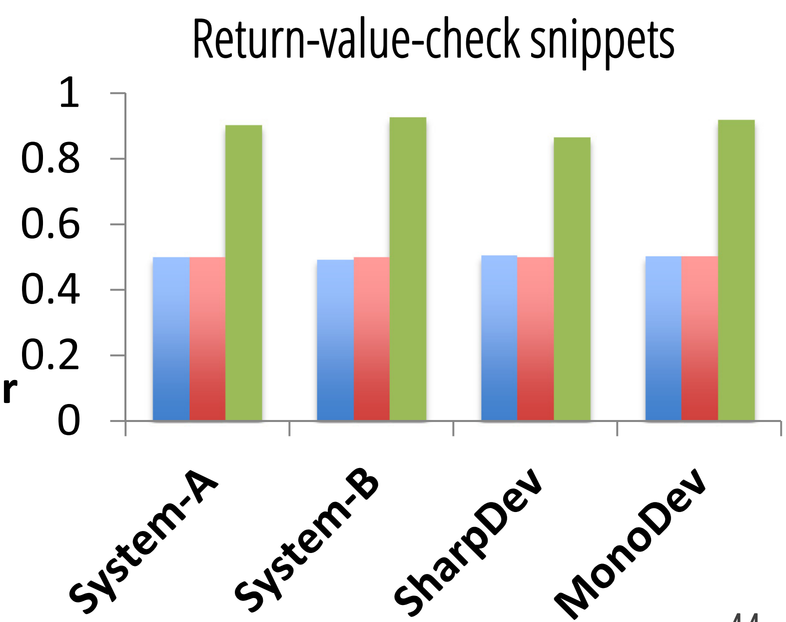
Across-project evaluation: one source project for training, one target project for testing

Evaluation (1)

Within-project evaluation

- Random: randomly logging (as a new developer)
- ErrLog [Yuan et al., OSDI'12]: logging all exception snippets
- LogAdvisor: BA results 0.846 ~ 0.934

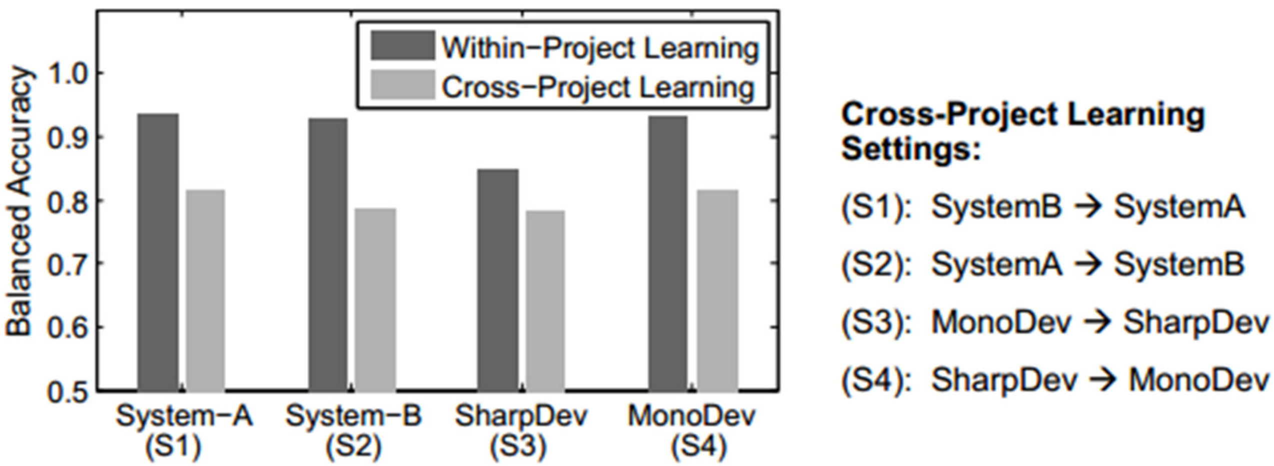




Evaluation (2)

Across-project evaluation

- Enrich the training data from other projects
- Extract common features among these projects
- BA results: above 0.8



Summary of Topic 1

- A "**learning to log**" framework aimed for automatic logging suggestion
- Evaluation on four large-scale software systems
  - **Industrial** systems and **open-source** systems
  - **Within-project** and **across-project** evaluation
- **Release of code and data** for future research:

  http://cuhk-cse.github.io/LogAdvisor
- Potential **impact in industry** (Microsoft)

Outline

- **Topic 1: Learning to log** for runtime service monitoring
- **Topic2: Online QoS prediction** of Web services
- Conclusion and future work

Outline

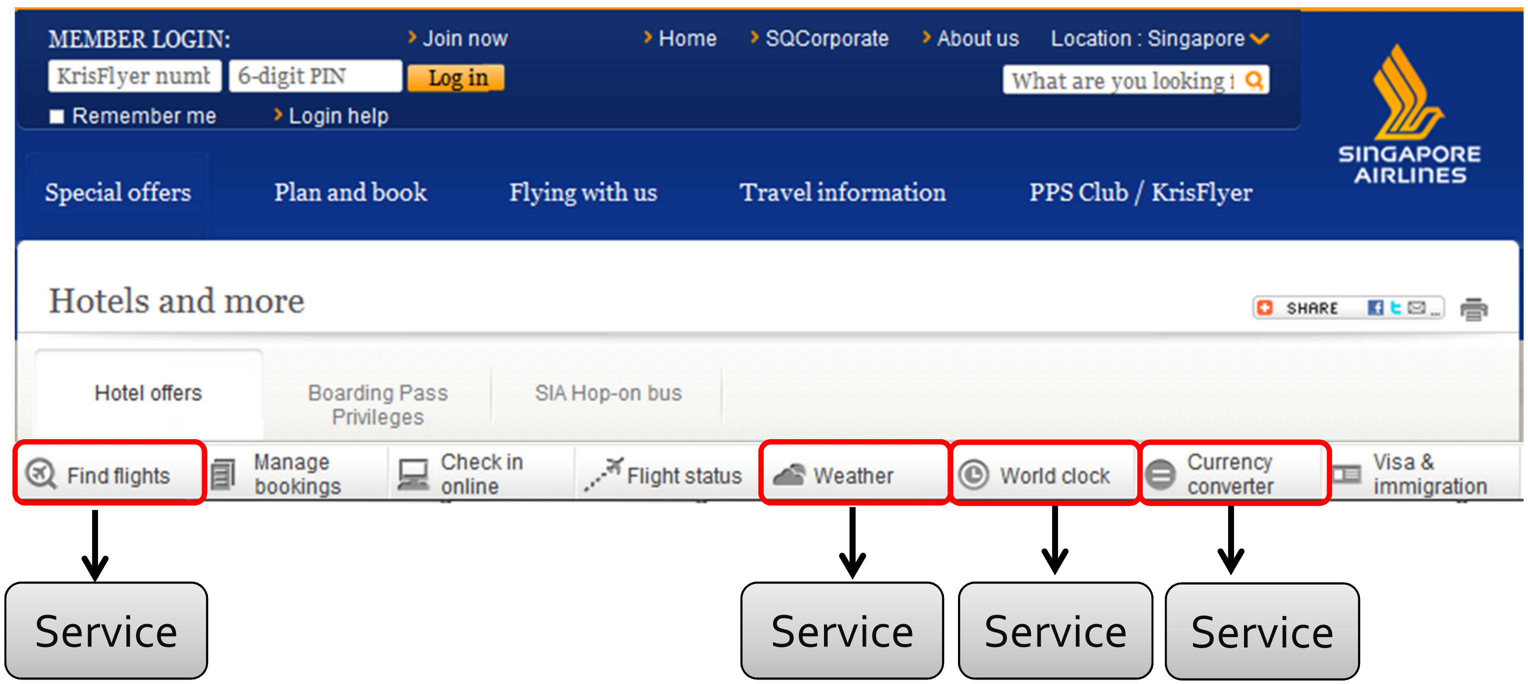
- **Topic2: Online QoS prediction** of Web services
  - Motivation
  - Adaptive matrix factorization
  - Experiments
  - Summary

Motivation

Web service: a component to build online services

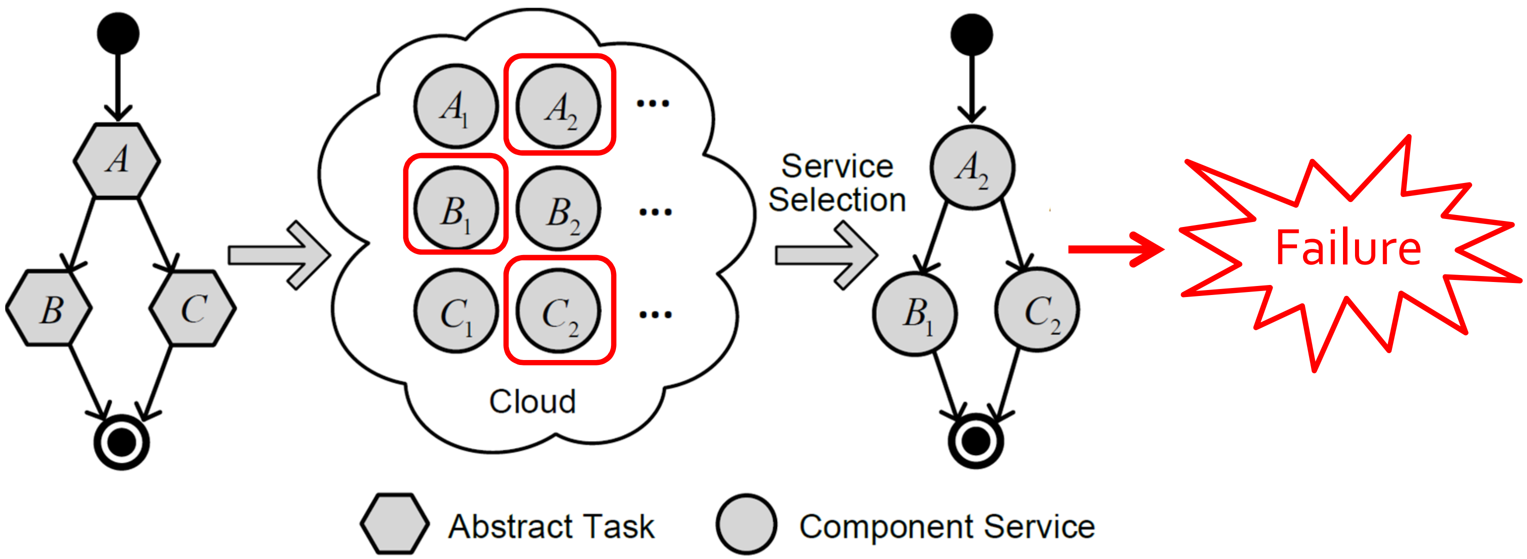
- Black-box (third-party) Web APIs
- Accessed over a network
- Executed on remote systems

Motivation

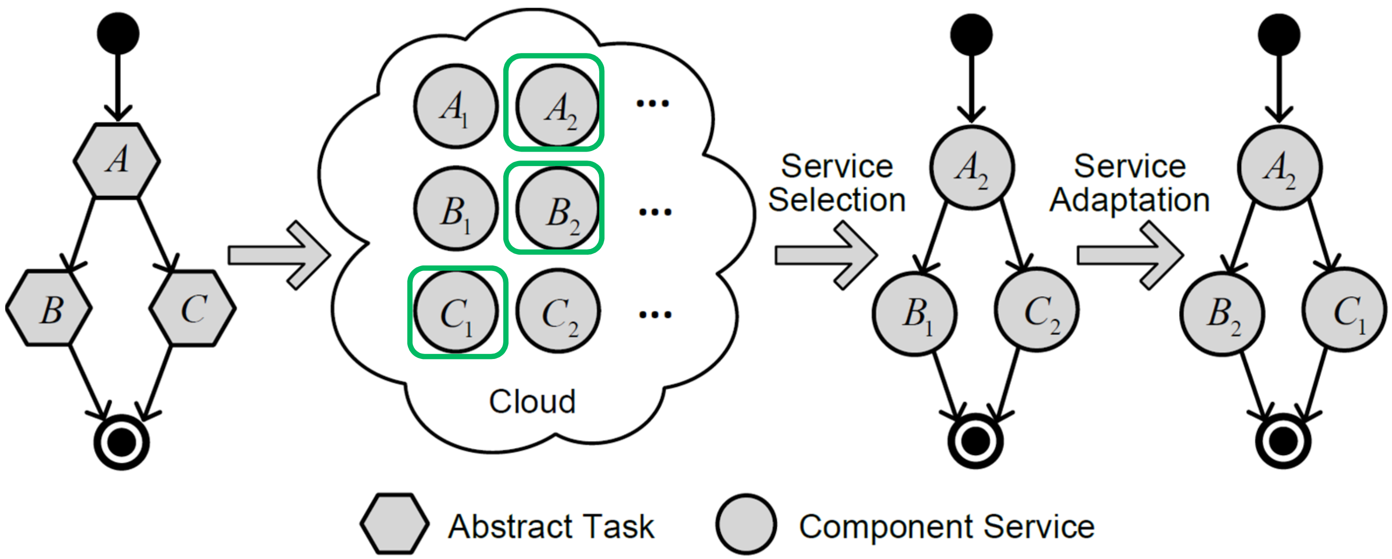


[Image from http://www.singaporeair.com]

Motivation


A
B
C
A_1
A_2
B_1
B_2
C_1
C_2
Cloud
Service Selection
A_2
B_1
C_2
Failure
Abstract Task
Component Service

Motivation



Runtime service adaptation:

[Moser et al. WWW'08][Cardellini et al., TSE'12]

switching a working service to a candidate service at runtime (e.g., B1 → B2, C2 → B1)

Motivation

Decisions for service adaptation:

- When to trigger an adaptation action?
- Which working services to be replaced?
- Which candidate services to employ?

Need **real-time** QoS information of services

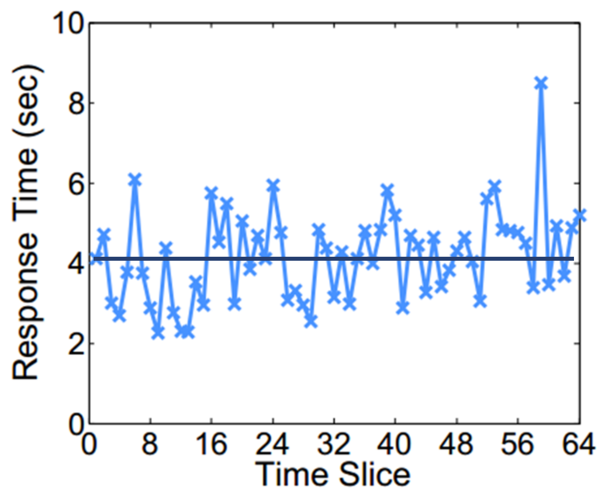
Motivation

Quality-of-Service (QoS)

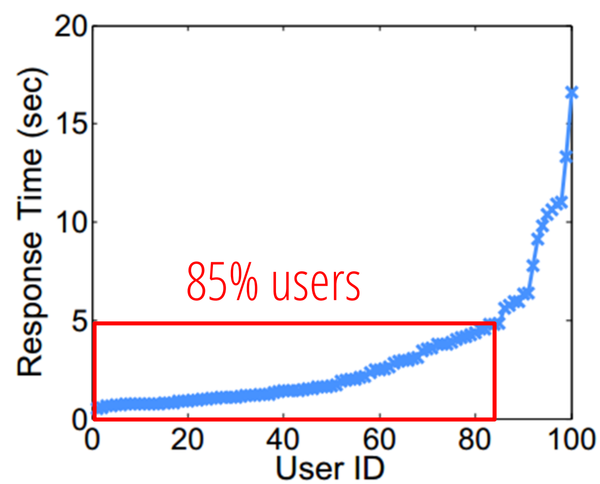
including response time, throughput, failure probability, etc.

– Time-varying
  - Dynamic network
  - Varying workload

– User-specific
  - Users distributed worldwide
  - Different networks



(a) RT v.s. Time Slice



(b) RT v.s. User ID

Motivation

Exhaustive measurement is infeasible

- Resource-consuming (large measurement overhead)
- Time-consuming (thousands of services)

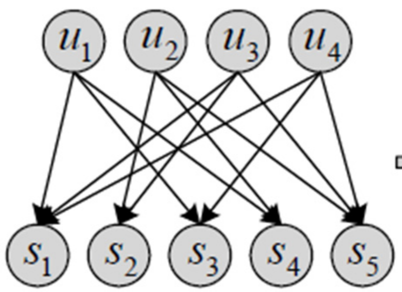
QoS prediction

by leveraging partial measurements to predict the remaining ones

- **Existing work**: e.g., monitoring or time-series based prediction for QoS of working services [Amin et al., ASE'12]
- **Unsolved problem**: QoS prediction of candidate services

Problem

The problem of Online QoS prediction



(a) User-Service Invocation Graph

	s_1	s_2	s_3	s_4	s_5
u_1	1.4	?	1.1	0.7	?
u_2	?	0.3	?	0.7	0.5
u_3	0.4	0.3	?	?	0.3
u_4	1.4	?	1.2	?	0.8

(b) Observed QoS Matrix

t_1 t_2 t_3

	s_1	s_2	s_3	s_4	s_5
u_1	1.4	0.8	1.1	0.7	0.9
u_2	1.0	0.3	1.0	0.7	0.5
u_3	0.4	0.3	0.3	0.1	0.3
u_4	1.4	0.7	1.2	0.8	0.8

(c) Dynamic QoS matrix

How to predict the unknown values **at runtime**?

Contributions of this work

AMF: adaptive matrix factorization

- An approach to enable **online**, **accurate**, and **scalable** QoS predictions

Key techniques

- Data transformation
- Online learning
- Adaptive weights

Outline

- **Topic2: Online QoS prediction** of Web services
  - Motivation
  - Adaptive matrix factorization
  - Experiments
  - Summary

Key observation

The measured QoS data matrix has an approximate low rank in nature

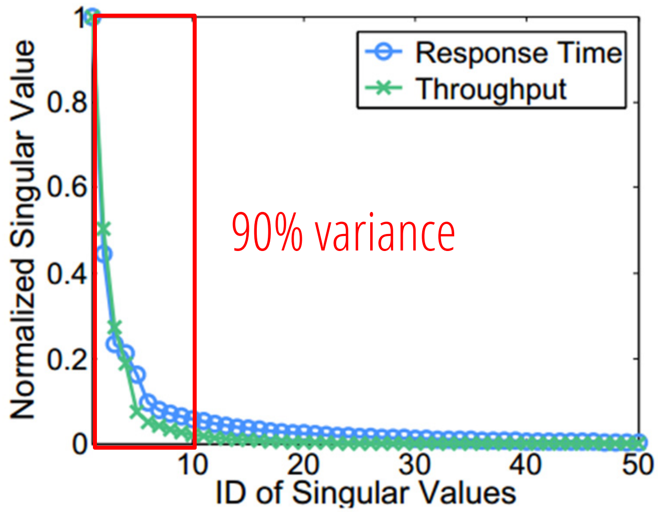


Fig. 9. Sorted Singular Values

Low-rank matrix approximation

Matrix factorization (MF): $R \approx U^T S$

	s_1	s_2	s_3	s_4	s_5
u_1	1.4	?	1.1	0.7	?
u_2	?	0.3	?	0.7	0.5
u_3	0.4	0.3	?	?	0.3
u_4	1.4	?	1.2	?	0.8

(b) Observed QoS Matrix

U^T

0.8	0.6
0.9	0.1
0.1	0.3
0.9	0.5

$\times$

S

1.0	0.2	1.0	0.8	0.4
1.0	1.0	0.5	0.1	0.9

$=$

	s_1	s_2	s_3	s_4	s_5
u_1	1.4	0.8	1.1	0.7	0.9
u_2	1.0	0.3	1.0	0.7	0.5
u_3	0.4	0.3	0.3	0.1	0.3
u_4	1.4	0.7	1.2	0.8	0.8

(c) Matrix Factorization

(d) Predicted QoS Matrix

Problem formulation:

$$\mathcal{L} = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{n} I_{ij}(R_{ij} - U_i^T S_j)^2 + \frac{\lambda_u}{2}\|U\|_F^2 + \frac{\lambda_s}{2}\|S\|_F^2$$

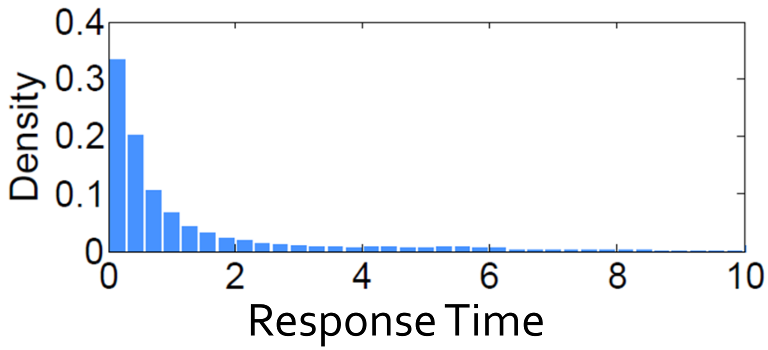
$$\left.\begin{aligned} U_i &\leftarrow U_i - \eta \sum_{j=1}^{n} I_{ij}(U_i^T S_j - R_{ij})(S_j) + \lambda_u U_i \\ S_j &\leftarrow S_j - \eta \sum_{i=1}^{m} I_{ij}(U_i^T S_j - R_{ij})(U_i^T) + \lambda_s S_j \end{aligned}\right\} \text{Gradient descent updates}$$

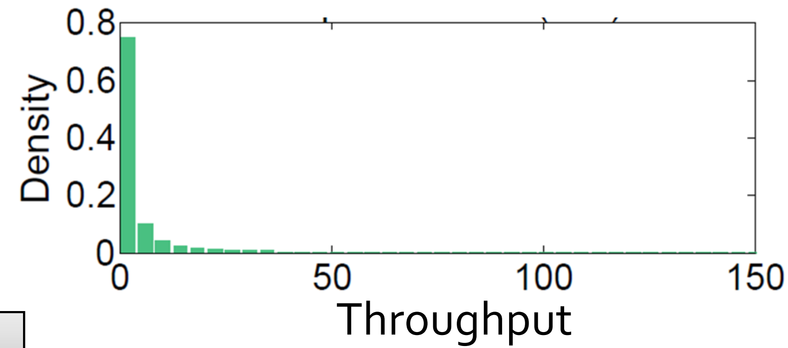
Challenges in applying MF to QoS prediction

- **Challenge 1**: skewed QoS value distributions
- **Challenge 2**: time varying QoS values
- **Challenge 3**: scalability on new users and services

Dealing with challenge 1

(skewed QoS distributions)

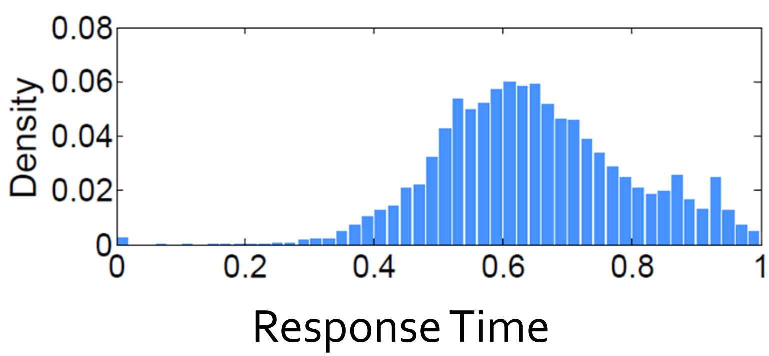


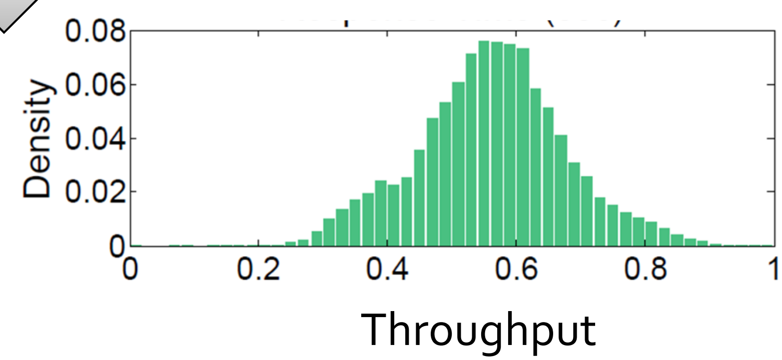


Box-Cox transformation

- Stabilize data variance
- Rank-preserving

$$boxcox(x) = \begin{cases} (x^\alpha - 1)/\alpha & \text{if } \alpha \neq 0\ , \\ \log(x) & \text{if } \alpha = 0, \end{cases}$$





Dealing with challenge 2

(time varying QoS values)

$$U_i \leftarrow U_i - \eta \sum_{j=1}^{n} I_{ij}(U_i^T S_j - R_{ij})(S_j) + \lambda_u U_i$$

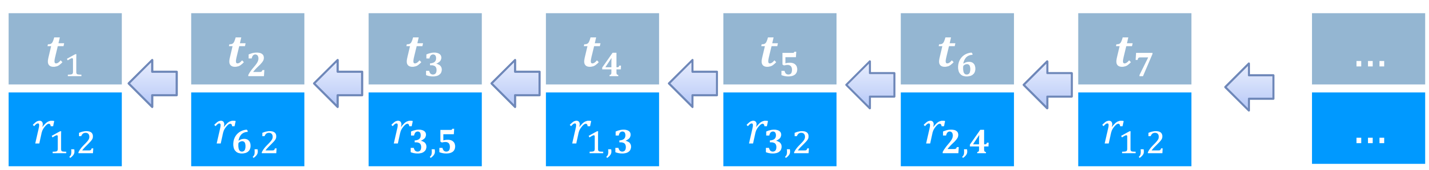
$$S_j \leftarrow S_j - \eta \sum_{i=1}^{m} I_{ij}(U_i^T S_j - R_{ij})(U_i^T) + \lambda_s S_j$$

Gradient descent works in batch mode

Online learning

- Stochastic gradient descent (SGD) algorithm
- Adapt to each newly observed data sample (u_i, s_j, R_{ij})

Updating in online mode:



SGD updating rules:

$$U_i \leftarrow U_i - \eta((g_{ij} - r_{ij})g'_{ij} S_j / r_{ij}^2 + \lambda_u U_i)$$

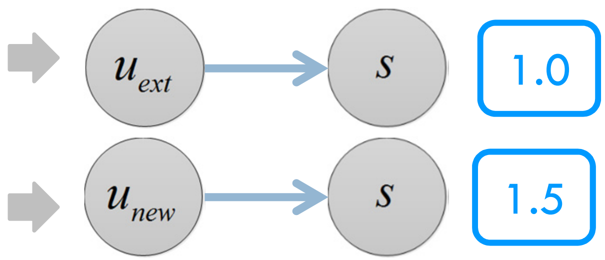
$$S_j \leftarrow S_j - \eta((g_{ij} - r_{ij})g'_{ij} U_i / r_{ij}^2 + \lambda_s S_j)$$

Dealing with challenge 3

(scalability on new users and services)

Adaptive weights

- **Weighted learning rate** for each user/service: Large for new vectors, small for converged vectors



Updating rules:

$$U_i \leftarrow U_i - \boxed{\eta w_{u_i}}((g_{ij} - r_{ij})g'_{ij}S_j/r_{ij}^2 + \lambda_u U_i)$$
$$S_j \leftarrow S_j - \boxed{\eta w_{s_j}}((g_{ij} - r_{ij})g'_{ij}U_i/r_{ij}^2 + \lambda_s S_j)$$

- Become robust
  - Existing users and services keep stable
  - New users and services converge fast

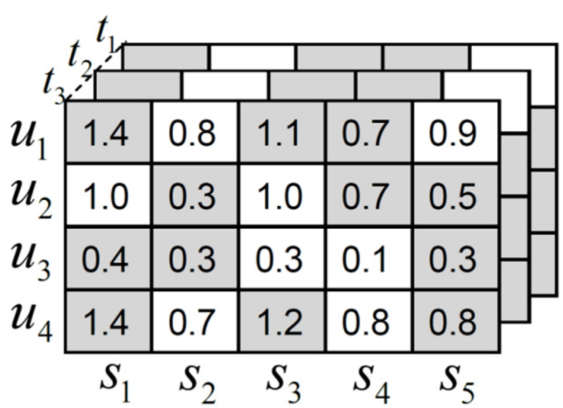
Outline

- **Topic2: Online QoS prediction** of Web services
  - Motivation
  - Adaptive matrix factorization
  - Experiments
  - Summary

Experiments

Data collection

- Response time (RT): user-perceived delay of a service invocation
- Throughput (TP): data transmission rate
- 142 * 4500 * 64 QoS matrix
  - 142 users (Planetlab nodes)
  - 4,500 real-world Web services
  - 64 time slices, at 15min time interval



Experiments

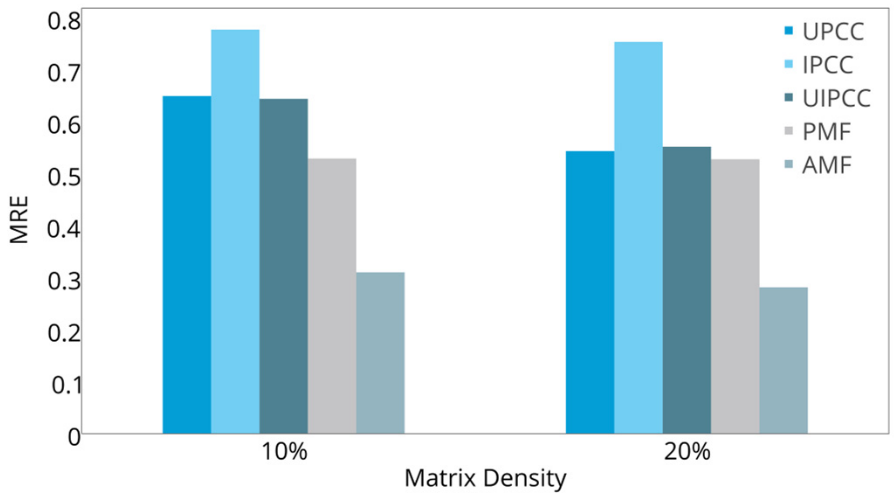
Evaluation metrics

- **MRE** (**median relative error**): 50% of the relative errors are below MRE
- **NPRE** takes the **90th percentile** of all the pairwise relative errors

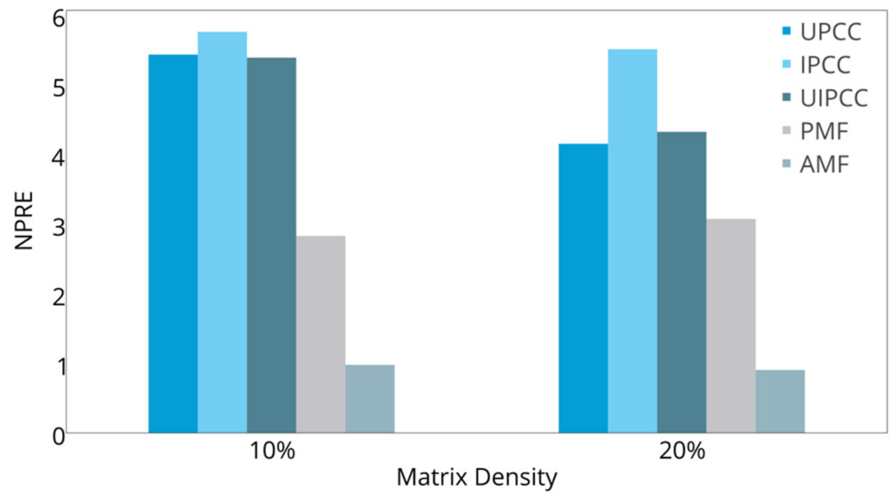
Baseline approaches to compare

- UPCC, IPCC, UIPCC: conventional collaborative filtering baselines
  [Shao et al., ICWS'07] [Zheng et al., ICWS'09][Zheng et al., TSC'11]
- PMF: convectional matrix factorization approach
  [Salakhutdinov et al, NIPS'07][Lo et al., SCC'12]
- These approaches **cannot perform online**

Response time results



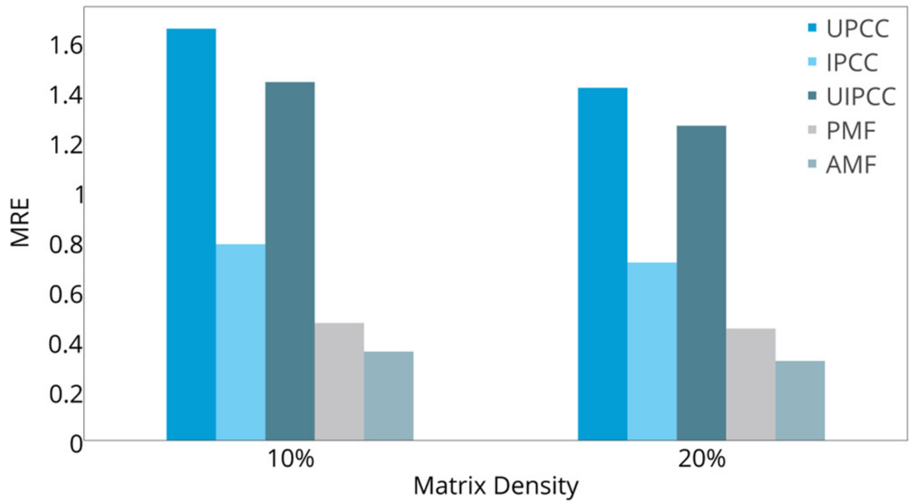
MRE



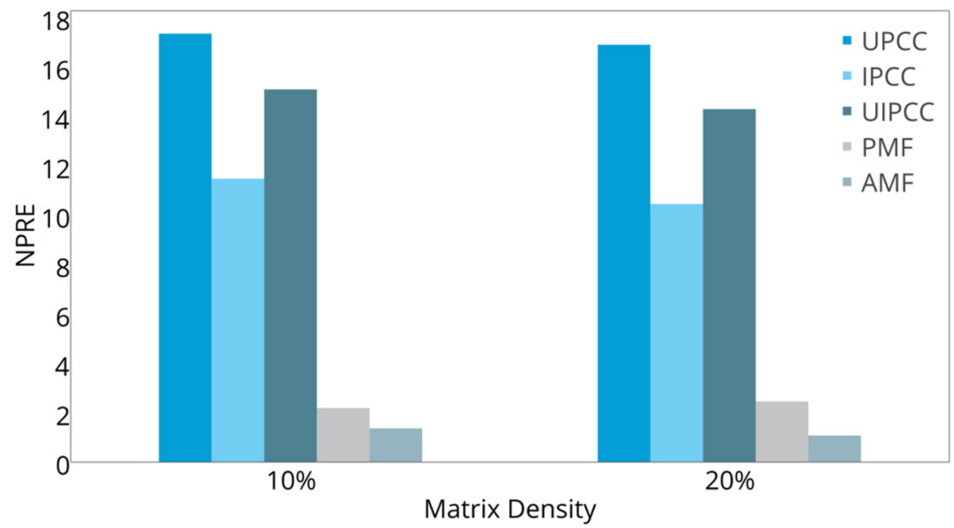
NPRE

AMF achieves **41%~46% improvement** in MRE, **65%~70% improvement** in NPRE

Throughput results



MRE



NPRE

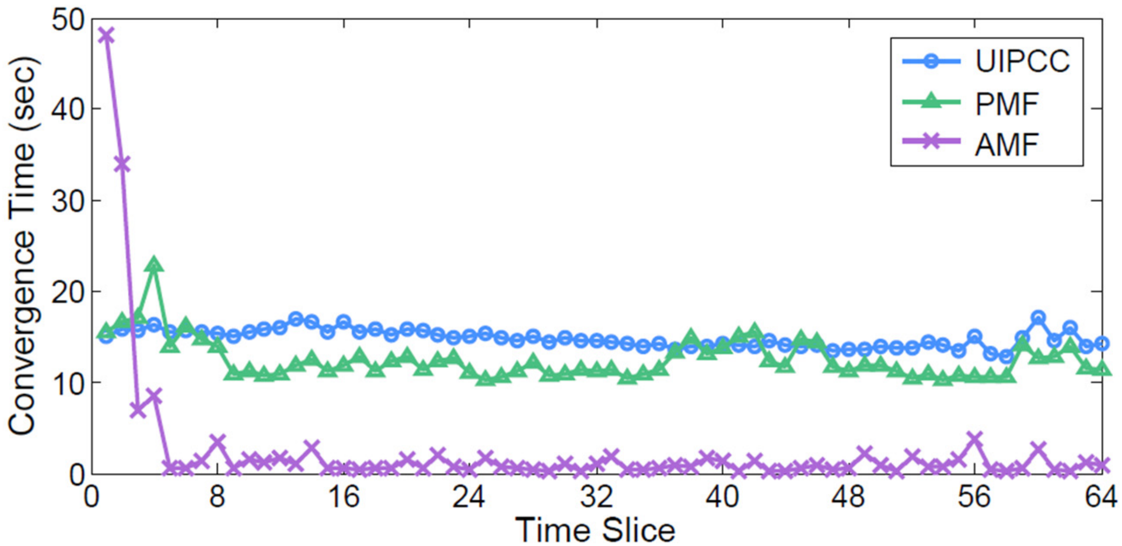
AMF achieves **24%~29% improvement** in MRE,
37%~56% improvement in NPRE

Efficiency analysis

Compared approaches

- UIPCC
- PMF

Re-train the entire model at each time slice



AMF: continuously and incremental updating

Summary of Topic 2

Online QoS prediction of Web services

– AMF: adaptive matrix factorization
– Techniques of data transformation, online learning, and adaptive weights
– Online, accurate, and scalable predictions

Release of code and datasets

– WS-DREAM dataset: http://www.wsdream.net

100+ downloads from 15 countries

– Code at Github: http://wsdream.github.io/AMF

Outline

- **Learning to log** for runtime service monitoring
- **Online QoS prediction** of Web services
- Conclusion and future work

Conclusion

Contributions

- **Learning to log** for runtime service monitoring
  - A framework to provide informative logging suggestions to developers
- **Online QoS prediction** of Web services
  - An online, accurate, and scalable QoS prediction approach

Conclusion

Contributions

- **Learning to log** for runtime service monitoring
  - A framework to provide informative logging suggestions to developers
- **Online QoS prediction** of Web services
  - An online, accurate, and scalable QoS prediction approach
- **Response time prediction** of Web services
  - A Web service positioning framework based on network coordinates
- **Privacy-preserving QoS prediction** of Web services
  - A privacy-preserving QoS prediction framework based on data randomization

Future work

Automatic logging

- **Where** to log vs **what** to log
- Tool support for developers

Massive log analysis

- To automate log analysis for failure diagnosis by using machine learning techniques

Publications (1)

1. **Jieming Zhu**, Pinjia He, Qiang Fu, Hongyu Zhang, Michael R. Lyu, and Dongmei Zhang. Learning to Log: Helping Developers Make Informed Logging Decisions. In *Proc. Of the International Conference on Software Engineering (ICSE)*, pages 415-425, 2015.
2. **Jieming Zhu**, Pinjia He, Zibin Zheng, and Michael R. Lyu. A Privacy-Preserving QoS Prediction Framework for Web Service Recommendation. In *Proc. of the IEEE International Conference on Web Services (ICWS)*, pages 241-248, 2015.
3. Cuiyun Gao, Baoxiang Wang, Pinjia He, **Jieming Zhu**, Yangfan Zhou, and Michael R. Lyu. PAID: Prioritizing App Issues for Developers by Tracking User Reviews Over Versions. In *Proc. of the IEEE International Symposium on Software Reliability Engineering (ISSRE)*, 2015.
4. **Jieming Zhu**, Pinjia He, Zibin Zheng, and Michael R. Lyu. Towards Online, Accurate, and Scalable QoS Prediction for Runtime Service Adaptation. In *Proc. of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, pages 318-327, 2014.
5. Qiang Fu, **Jieming Zhu**, Wenlu Hu, Jian-Guang Lou, Rui Ding, Qingwei Lin, Dongmei Zhang, and Tao Xie. Where Do Developers Log? An Empirical Study on Logging Practices in Industry. In *Proc. of the International Conference on Software Engineering (ICSE)*, pages 24-33, 2015.
6. Pinjia He, **Jieming Zhu**, Zibin Zheng, Jianlong Xu, and Michael R. Lyu. Location-based Hierarchical Matrix Factorization for Web Service Recommendation. In *Proc. of the IEEE International Conference on Web Services (ICWS)*, pages 297-304, 2014.

Publications (2)

7. Pinjia He, **Jieming Zhu**, Jianlong Xu, and Michael R. Lyu. A Hierarchical Matrix Factorization Approach for Locationbased Web Service QoS Prediction. In *Proc. of the International Workshop on Internet-based Virtual Computing Environment (iVCE)*, pages 290-295, 2014.
8. **Jieming Zhu**, Zibin Zheng, and Michael R. Lyu. DR2: Dynamic Request Routing for Tolerating Latency Variability in Online Cloud Applications. In *Proc. of the IEEE International Conference on Cloud Computing (CLOUD)*, pages 589-596,2013.
9. Zibin Zheng, **Jieming Zhu**, and Michael R. Lyu. Service-generated Big Data and Big Data-as-a-Service: An Overview. In *Proc. of the IEEE International Congress on Big Data*, pages 403-410, 2013.
10. **Jieming Zhu**, Zibin Zheng, Yangfan Zhou, and Michael R. Lyu. Scaling Service-oriented Applications into Geo-distributed Clouds. In *Proc. of the International Workshop on Internetbased Virtual Computing Environment (iVCE)*, pages 335-340, 2013.
11. **Jieming Zhu**, Yu Kang, Zibin Zheng, and Michael R. Lyu. WSP: A Network Coordinate based Web Service Positioning Framework for Response Time Prediction. In *Proc. of the IEEE International Conference on Web Services (ICWS),* pages 90-97, 2012.
12. **Jieming Zhu**, Yu Kang, Zibin Zheng and Michael R. Lyu. A Clustering-based QoS Prediction Approach for Web Services Recommendation. In *Proc. of the International Workshop on Internet-based Virtual Computing Environment (iVCE)*, pages 93-98, 2012.

Thank you!

Q&A

FAQ1: Learning to log

1. How many logging statements are there in your studied systems ? And what's the logging ratio in the code?
2. What is the effect of different machine learning models?
3. What is the effect of imbalance handling?
4. Why do you use Balanced Accuracy for evaluation? Why not precision and recall?
5. Why not evaluate your LogAdvisor tool with real developers?
6. What are the factors to determine whether to log or not in practice?

FAQ2: Learning to log

7. You said logging is pervasive. Why did I not write logging code at all?
8. Exceptions occur occasionally. Why not log them all? What will happen?
9. Why did you only study systems written in C# ? Can LogAdvisor be applied to systems in other languages?
10. LogAdvisor learns from existing code. What if the project has bad logging practice?
11. Sounds good. Are there any limitations?
12. Is this work industry-driven? Or is it a one off paper?
13. I totally don't get why you are doing this!?

FAQ3: Online QoS prediction

1. What is the impact of data transformation on accuracy?
2. How did you evaluate the scalability of AMF?
3. What is the impact of matrix density on accuracy?
4. What is the main difference between AMF and MF?
5. Why is MRE (relative error) better than MAE (absolute error) in evaluation?
6. What is the main purpose of adaptive weights? How to assign them?
7. What is the approach of UIPCC?
8. How can we use AMF prediction results for runtime service adaptation?