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...
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

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

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.
Quality management of online service systems is important, but challenging.
Online service systems are becoming large-scale in size and complex in structure.
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]
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

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

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

Online QoS prediction [ICDCS’14] (Chapter 4)

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

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

Logging too much

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

*User:
“Apache httpd cannot start. No log message printed.”*

[Yuan et al., OSDI’12]
Challenges of logging

Logging too little

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

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

Logging too much

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

1. Instances Collection
   - Software Repositories
   - Focused Code Snippets
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
Focused snippets: indicate potential error sites
  — Exception snippets: try-catch blocks
  — Return-value-check snippets: function-return errors

Exception snippet example

```java
try {
    method(...);
} catch (IOException) {
    log(...);
}...
```

Return-value-check snippet example

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

```csharp
try {
    method(...);
} catch (IOException) {
    log(...);
...
}
```

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

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

**Contextual feature extraction**

- Structural features
- Textual features
- Syntactic features
Feature extraction (1)

**Structural features:** structural info of code

```csharp
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. */
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; }
    ...
}
Feature extraction (3)

**Syntactic features:** syntactic info of code

```csharp
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 rulesFrom assembly '{0}'.", assembly); return 0; }
    ...
}
```
<table>
<thead>
<tr>
<th>(1) Instances Collection</th>
<th>(2) Label Identification</th>
<th>(3) Feature Extraction</th>
<th>(4) Feature Selection</th>
<th>(5) Model Construction</th>
<th>(6) Logging Suggestion</th>
</tr>
</thead>
</table>

### Feature selection

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

- Remove infrequency 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
• **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)

\[
BA = \frac{1}{2} \times \frac{TP}{TP + FN} + \frac{1}{2} \times \frac{TN}{TN + FP}
\]

- Accuracy of logged instances
- 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

• **Topic 2:** Online QoS prediction of Web services

• Conclusion and future work
Outline

• **Topic 2: 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
Motivation
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., $B_1 \rightarrow B_2$, $C_2 \rightarrow B_1$)
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

85% users
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

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

90% variance
Low-rank matrix approximation

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

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

Gradient descent updates

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

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

\[
\text{boxcox}(x) = \begin{cases} 
  \frac{x^\alpha - 1}{\alpha} & \text{if } \alpha \neq 0, \\
  \log(x) & \text{if } \alpha = 0,
\end{cases}
\]
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:

\[ U_i \leftarrow U_i - \eta((g_{ij} - r_{ij})g_{ij}U_i/r_{ij}^2 + \lambda_u U_i) \]
\[ S_j \leftarrow S_j - \eta((g_{ij} - r_{ij})g_{ij}U_i/r_{ij}^2 + \lambda_s S_j) \]
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:

\[ U_i \leftarrow U_i - \eta w_{u_i} ((g_{ij} - r_{ij})g_{ij}^2 S_j / r_{ij}^2 + \lambda_u U_i) \]
\[ S_j \leftarrow S_j - \eta w_{s_j} ((g_{ij} - r_{ij})g_{ij}^2 U_i / r_{ij}^2 + \lambda_s S_j) \]

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

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


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?