From the Outside Looking In: Probing Web APIs to Build Detailed Workload Profile

Nan Deng, Zichen Xu, Christopher Stewart and Xiaorui Wang
The Ohio State University
The typical web page loads data from 7-25 third party providers [Everts, 2013]

In 2013, the number of indexed APIs grew by 32% year over year [PW, 2013]
From the Outside Looking In

• Using Web APIs
  – Improve content without programming
  – Published interfaces provide well defined, often RESTful, output
  – Data is centralized, managed by experts

• Benefits
  – Salaries are 20% of expenses [tripAdvisor]
  – Failures, dynamic workloads, corner cases covered
  – Efficient to move compute to big data

1. Motivation
2. Problem
3. Our Approach
Using web APIs risks availability & performance

- “Everyone has bad days, and third-party content providers are no exception.” Tammy Everts

- Sometimes web APIs perform poorly because they were implemented poorly

- “…a bug [affected] people from third party sites integrated with Facebook” on Feb 7, 2013
  Took down CNN & WaPost

- CDN Planet Homepage reported that Facebook took 796 ms to load, 2X longer than any other critical content
  Slow responses cost 3.5B/y [Everts, 2013]
Is the web API well implemented?
How will it respond under extreme operating conditions?
Challenge: Create useful profiles faster than trial-and-error approach
Hypothesis: Given cloud design, response times carry a lot of valuable information.
From the Outside Looking In

1. Motivation
2. Problem
3. Our Approach

- **Use Case: Planning web API Usage**
- Google Maps versus Bing Maps
  - Same avg. resp. time & availability
  - Which has heavier tail?
  - Should we use both (Replication for predictability [Fast 10])

- **Use case: Model Resource Needs of Third Parties**
  - DataGreening and Ecosia are green hosts [ICAC 2012]
    - DataGreening offsets the carbon footprint of email users that route through its servers
  - Must model carbon footprint of IMAP web APIs
From the Outside  
Looking In

1. Motivation
2. Problem
3. Our Approach

- Related work and alternative approaches
  - Controlled offline tests yield workload profiles
    [ugaonkar,2005][stewart,2005]
  - Tracing online execution of requests
    [isaacs,2004][shen,2008][PowerTracer,ICAC]
  - Use logs from online execution to infer profiles
    [stewart,2008]
- More “inside” access than web API permit
  - web API encouraged to hide details and provide false data
Extracting Workload Profiles

1. Cloud Practices
2. ICA
3. Early Results

• Widely used cloud computing practices
  Structure of cloud-based web API
  – Hierarchical tiered design
  – **Elastic scaling**
  – **Make the common case fast**

• Independent component Analysis
  – **Application to workload profiling**

• Early results
Extracting Workload Profiles

1. Cloud Practices
2. ICA
3. Early Results

- **Elastic scaling**: When resource demands increase, provision more resources. When demands decrease, release resources.

- Active research: React to metrics other than response time to better stabilize performance without using too many resources
  
  [Ghandi, TOC 2012] [Nguyen, ICAC 2013]

- **Response time is less sensitive to workload changes**
Extracting Workload Profiles

1. Cloud Practices
2. ICA
3. Early Results

- Make the common case fast [P & H]
  - In software design: Data processing in background
  - In platform design: Garbage collection not on critical path
  - In hardware design: Guard band prevents 99.99% of timing errors that would otherwise trigger ECC in processor cache

- **Response times follow skewed multi-modal distributions**

  ![Graph](https://via.placeholder.com/150)

  99th % > 99X mean

  3-node Zookeeper on 4 core
  2.4Ghz, data size = 1 GB,
  100K writes issued serially

  Graph and data from Stewart et al. ICAC 13
Extracting Workload Profiles

1. Cloud Practices
2. ICA
3. Early Results

- Blind source separation techniques infer source signals from output signals
  - Given $F(X,Y,Z)$ infer $X$, $Y$ and/or $Z$

- Independent Component Analysis (ICA) is an established approach [Herault & Jutton, 1986] provided sources are Independent & Non-Gaussian
ICA is famously used to recover audio signals

Example: 3 people sing their favorite song at the same time. They stand still in the same room. 3 microphones record output.
ICA is famously used to recover audio signals.

<table>
<thead>
<tr>
<th>Time</th>
<th>Mic #1</th>
<th>Mic #2</th>
<th>Mic #3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>2</td>
<td>6</td>
</tr>
</tbody>
</table>

What is the least Gaussian signal that could have produced this data?
Key insight: Use ICA to infer web API components
- Transform spatial dimension into concurrency

Response time is the sum of service time and queuing delay of comp. on the critical path
Key insight: Use ICA to infer web API components
- Transform spatial dimension into concurrency
- Audio sources to component service times
ICA captures service time distribution of web API components provided

- Service times are non-Gaussian (Common case fast)
- Service times are independent (Elastic scaling)

Final output is normalized CDF for each component
Extracting Workload Profiles

- Benchmark comprised of widely used components
  - Serving non-stationary request arrivals [stewart, 2007]
- 3 components on the critical path
Extracting Workload Profiles

- Hand matched normalized CDF to source components
  - Less than 5% prediction error
- Setup the same benchmark using Zookeeper as a cache instead of Redis
  - Zookeeper is a poor choice because each insert involves Paxos and capacity per cluster is lower than Redis

Zookeeper setup had the fatter tail.

This could be a warning sign.
Conclusion

- Web API hide workload details that could help Internet services plan for performance

- Programming practices in cloud computing allow new inferences about workloads

- Blind source separation techniques yield useful workload profiles within the web API model
  - From the outside looking in, we can infer a lot