Shadow: The Case for Determinism on the Edge

Matthew Furlong, Andrew Quinn, and Jason Flinn
**Concerns for the Edge**

<table>
<thead>
<tr>
<th>Edge Concern</th>
<th>Effect</th>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Concerns for the Edge

<table>
<thead>
<tr>
<th>Edge Concern</th>
<th>Effect</th>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provisioning</td>
<td>More jitter</td>
<td></td>
</tr>
</tbody>
</table>
## Concerns for the Edge

<table>
<thead>
<tr>
<th>Edge Concern</th>
<th>Effect</th>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provisioning</td>
<td>More jitter</td>
<td>• Limited edge resources</td>
</tr>
</tbody>
</table>
## Concerns for the Edge

<table>
<thead>
<tr>
<th>Edge Concern</th>
<th>Effect</th>
<th>Problem</th>
</tr>
</thead>
</table>
| Provisioning | More jitter | • Limited edge resources  
|              |          | • Mobile network variability               |

![Graph showing CDF of latency (RTT in msec) for different networks](image)

High tail latency!

## Concerns for the Edge

<table>
<thead>
<tr>
<th>Edge Concern</th>
<th>Effect</th>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provisioning</td>
<td>More jitter</td>
<td>• Limited edge resources</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mobile network variability</td>
</tr>
<tr>
<td>Mobility</td>
<td>Performance variability</td>
<td></td>
</tr>
</tbody>
</table>
## Concerns for the Edge

<table>
<thead>
<tr>
<th>Edge Concern</th>
<th>Effect</th>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provisioning</td>
<td>More jitter</td>
<td>• Limited edge resources&lt;br&gt;• Mobile network variability</td>
</tr>
<tr>
<td>Mobility</td>
<td>Performance variability</td>
<td>• Low-latency connections are lost</td>
</tr>
</tbody>
</table>
## Concerns for the Edge

<table>
<thead>
<tr>
<th>Edge Concern</th>
<th>Effect</th>
<th>Problem</th>
</tr>
</thead>
</table>
| Provisioning | More jitter | • Limited edge resources  
• Mobile network variability |
| Mobility     | Performance variability | • Low-latency connections are lost |
| Reliability  | More faults |  |
## Concerns for the Edge

<table>
<thead>
<tr>
<th>Edge Concern</th>
<th>Effect</th>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provisioning</td>
<td>More jitter</td>
<td>• Limited edge resources</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mobile network variability</td>
</tr>
<tr>
<td>Mobility</td>
<td>Performance</td>
<td>• Low-latency connections are lost</td>
</tr>
<tr>
<td></td>
<td>variability</td>
<td></td>
</tr>
<tr>
<td>Reliability</td>
<td>More faults</td>
<td>• Numerous providers</td>
</tr>
</tbody>
</table>
## Concerns for the Edge

<table>
<thead>
<tr>
<th>Edge Concern</th>
<th>Effect</th>
<th>Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provisioning</td>
<td>More jitter</td>
<td>• Limited edge resources</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Mobile network variability</td>
</tr>
<tr>
<td>Mobility</td>
<td>Performance variability</td>
<td>• Low-latency connections are lost</td>
</tr>
<tr>
<td>Reliability</td>
<td>More faults</td>
<td>• Numerous providers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Distributed</td>
</tr>
</tbody>
</table>
## Concerns for the Edge

<table>
<thead>
<tr>
<th>Edge Concern</th>
<th>Effect</th>
<th>Problem</th>
</tr>
</thead>
</table>
| Provisioning       | More jitter                     | • Limited edge resources  
|                    |                                 | • Mobile network variability                               |
| Mobility           | Performance variability         | • Low-latency connections are lost                         |
| Reliability        | More faults                     | • Numerous providers  
|                    |                                 | • Distributed                                              |
| More data sources  | Privacy concerns                |                                                            |


### Concerns for the Edge

<table>
<thead>
<tr>
<th>Edge Concern</th>
<th>Effect</th>
<th>Problem</th>
</tr>
</thead>
</table>
| Provisioning          | More jitter             | • Limited edge resources  
                       |                                                        | • Mobile network variability                     |
| Mobility              | Performance variability | • Low-latency connections are lost                         |
| Reliability           | More faults             | • Numerous providers  
                       |                                                        | • Distributed                                    |
| More data sources     | Privacy concerns        | • More IoT devices = more sensitive user data              |
Applications are typically latency-sensitive or require high bandwidth
Moving From the Cloud to Edge

- Applications are typically latency-sensitive or require high bandwidth
Outline

• Motivation
• Determinism to the rescue
• Shadow: Deterministic containers
• Evaluation
A deterministic execution of an application provides the necessary support for edge systems.
Determinism to the Rescue

- Determinism allows applications to be replicated across multiple servers
Determinism to the Rescue

- Determinism allows applications to be replicated across multiple servers
- Determinism enables migration with minimal downtime
Migration

Better connection discovered

Server inputs: [a]

Server Responses: 1
Initiate migration!
Migration

Server inputs: a b

Checkpoint Application

Server inputs: a

Server Responses: 1 ?
Migration

Server inputs: a b

Transfer
Checkpoint

Multicast client input

Server Responses: 1 2

Server inputs: a b
Migration

Server inputs: a b c Server inputs: a b c

Transfer Checkpoint

Multicast client input

Server Responses: 1 2 3
Migration

Server inputs: \[a \ b \ c \ d\]

Multicast client input

Server responses: \[1 \ 2 \ 3 \ 4\]

Server inputs: \[a \ b \ c \ d\]

Restore Application

Server inputs: \[a \ b \ c \ d\]

Multicast client input

Server responses: \[1 \ 2 \ 3 \ 4\]
Migration

Server inputs: a b c d e

Server inputs: a b c d e

Multicast client input

Server Responses: 1 2 3 4 5

e e e
Migration

Server inputs: a b c d e f

Multicast client input

Server inputs: a b c d e f

Server Responses: 1 2 3 4 5 6
Migration

Server inputs: a b c d e f | Server inputs: a b c d e f

Migration complete!

Client inputs: 1 2 3 4 5 6
## Determinism to the Rescue

<table>
<thead>
<tr>
<th>Edge Concern</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
</table>
| Provisioning  | • Limited edge resources  
                • Mobile network variability                                      | Determinism |
| Mobility       | • Low-latency connections are lost                                     | Determinism |
| Reliability    | • Numerous providers  
                • Distributed                                                          | Determinism |
| More data sources | • More IoT devices = more sensitive user data                        | Determinism |
## Determinism to the Rescue

<table>
<thead>
<tr>
<th>Edge Concern</th>
<th>Problem</th>
<th>Solution</th>
</tr>
</thead>
</table>
| Provisioning  | • Limited edge resources  
• Mobile network variability                                   | Determinism |
| Reliability   | • Numerous providers  
• Distributed                                                     | Determinism |
| More data sources | • More IoT devices = more sensitive user data                 | Determinism |

**Challenge:** Applications are inherently non-deterministic!
Determinism to the Rescue

- Non-determinism in applications
  - Randomness
  - Time
  - Multi-threading
  - Network communication
  - Inter-process communication (IPC)
Non-determinism in applications
- Randomness
- Time
- Multi-threading
- Network communication
- Inter-process communication (IPC)

Challenge: How can we make non-deterministic applications behave deterministically??
Outline

• Motivation
• Determinism to the rescue
• Shadow: Deterministic containers
• Evaluation
• Goal: Eliminate sources of non-determinism
• Goal: Eliminate sources of non-determinism
  • Minimize application modifications
• Goal: Eliminate sources of non-determinism
  • Minimize application modifications
  • Maintain similar performance to non-deterministic executions
Deterministic Containers

- Shadow removes non-determinism from applications deployed within containers
  - Multi-threading
  - OS Events
  - Client inputs
  - External Sources of Non-Determinism
Shadow uses a deterministic thread scheduler based on *weak* determinism\(^1\)

• Shadow uses a deterministic thread scheduler based on *weak* determinism

• Weak determinism globally orders all synchronization operations

Deterministic Containers

- Shadow removes non-determinism from applications deployed within containers
  - Multi-threading
  - OS Events
  - Client inputs
  - External Sources of Non-Determinism
Operating System Events

• System Calls
  • getrandom()
  • gettimeofday()
  • getpid()
  • ...
  • ...
Deterministic Containers

- Shadow removes non-determinism from applications deployed within containers
  - Multi-threading
  - OS Events
  - Client inputs
  - External Sources of Non-Determinism
Client Inputs

- Determinism is only useful if replicas receive the same sequence of inputs
Determinism is only useful if replicas receive the same sequence of inputs.

Client multicasts inputs.
Determinism is only useful if replicas receive the same sequence of inputs.

Inputs are delivered in *epochs* to each server.
External Sources

- Determinism is only useful if replicas receive the same sequence of inputs
External Sources

- Determinism is only useful if replicas receive the same sequence of inputs

3rd party service responses are not identical
Determinism is only useful if replicas receive the same sequence of inputs.
External Sources

- Determinism is only useful if replicas receive the same sequence of inputs

Forward request from client
Determinism is only useful if replicas receive the same sequence of inputs.
Determinism is only useful if replicas receive the same sequence of inputs.
Determinism is only useful if replicas receive the same sequence of inputs.
Determinism is only useful if replicas receive the same sequence of inputs.
Determinism is only useful if replicas receive the same sequence of inputs.

3rd party services can produce non-deterministic results and still work with Shadow!
# Determinism Summary

<table>
<thead>
<tr>
<th>Non-Determinism</th>
<th>Solution</th>
<th>Application Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-threading</td>
<td>Weak determinism</td>
<td>No data races</td>
</tr>
<tr>
<td>OS Events</td>
<td>Deterministic syscalls</td>
<td>Time skew No IPC or kernel devices</td>
</tr>
<tr>
<td>Client Inputs</td>
<td>Use client as primary</td>
<td>None</td>
</tr>
<tr>
<td>External Sources</td>
<td>Use client as primary</td>
<td>None</td>
</tr>
</tbody>
</table>
Outline

• Motivation
• Determinism to the rescue
• Shadow: Deterministic containers
• Evaluation
Evaluation

• Test Application
  • Vehicle Caravan
• Emulate suburban network traces from VNperf
• Evaluate quality of experience for the client using response time from server

Jitter Results

Vehicle Caravan

Cumulative Probability

Latency for Client (RTT in ms)
Jitter Results

Vehicle Caravan

Shadow reduces the median response by up to 25 percent
Discussion

- Feedback
  - Are Shadow’s limitations reasonable to place on applications?
  - How prevalent does the community think single-client applications will be in edge deployments?
  - What other applications could benefit from a deterministic environment?

- Controversial points
  - Applications must fit into Shadow’s restrictions (e.g. no IPC, clock skew, race-free)

- Everything falls apart…
  - Performance overhead of determinism outweighs the benefits
  - If the majority of applications cannot be modified to fit into Shadow’s restrictions