Active-Standby for High-Availability in FaaS

*Presented at: Sixth International Workshop on Serverless Computing (WoSC6) 2020*

Yasmina BOUIZEM
Inria, University of Rennes, France
LRIT, University of Tlemcen, Algeria

Christine MORIN, Inria, France
Djawida DIB, University of Tlemcen, Algeria
Nikos PARLAVANTZAS, IRISA, INSA Rennes, France

https://www.serverlesscomputing.org/wosc6/#p6
Function-as-a-Service (FaaS)
Objective

- Develop a solution to achieve High-Availability (HA) in FaaS

- Proposed a HA approach for FaaS based on active standby
- Implemented approach in an open-source FaaS platform, namely Fission
- Compared with the retry-based approach
Fission Architecture
Retry Mechanism in Fission

Fission Vanilla
Active-Standby Approach
Implementation in Fission

Fission Active-Standby (AS)

Executor
- New deploy

Service
- Standby-Pod (Standby-IP)
- Active-Pod (Active-IP)

Kubernetes

CoreDNS

1. Resolve A “Service”
2. A Active-IP
3. Request
4. Response
Experimental Setup

- **FaaS Frameworks**
  - Fission vanilla
  - Fission AS

- **Workload**
  - 3000 requests during 5 minutes

- **Fault Injection**
  - Function pod failure at a random time between 30 s and 60 s
  - Node crash 30 s after the beginning of the workload execution

- **Environment Setup**
  - Functions: Fibonacci.py, Guestbook.py
  - 7 nodes (5 for the cluster, 1 to invoke functions and another to inject faults)
  - Scenarios:
    - Pod failure
    - Node failure

- **Metrics**
  - Throughput
  - Response Time
  - Recovery Time
Performance Results: (1) Pod Failure

**Guestbook Application**

**Throughput**

- **Requests rate (r/sec)**
- **Time (sec)**

**Response Time**

- **Requests duration (msec)**
- **Time (sec)
User Perception: (1) Pod Failure

Fission Vanilla

HTTP Code Response rate

Fission AS

HTTP Code Response rate
## Availability Results: (1) Pod Failure

### Recovery Time

<table>
<thead>
<tr>
<th>Guestbook Application</th>
<th>Fission Vanilla</th>
<th>Fission AS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3.614s</td>
<td>1.528s</td>
</tr>
</tbody>
</table>
Performance Results: (2) Node Failure

Guestbook Application

Throughput

Response Time

Requests rate (r/sec)

Requests duration (msec)

Time (sec)
User Perception: (2) Node Failure

**Fission Vanilla**

HTTP Code Response rate

<table>
<thead>
<tr>
<th>Code</th>
<th>Number/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td></td>
</tr>
<tr>
<td>503</td>
<td></td>
</tr>
<tr>
<td>502</td>
<td></td>
</tr>
</tbody>
</table>

**Fission AS**

HTTP Code Response rate

<table>
<thead>
<tr>
<th>Code</th>
<th>Number/sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td></td>
</tr>
</tbody>
</table>

Time (sec)
## Availability Results: (2) Node Failure

### Recovery Time

<table>
<thead>
<tr>
<th></th>
<th>Fission Vanilla</th>
<th>Fission AS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Guestbook Application</td>
<td>2min39s</td>
<td>6.194s</td>
</tr>
</tbody>
</table>
Conclusion & Future Work

➢ Experiments showed that the Active-Standby approach outperforms the one based on the Retry mechanism in terms of response time and availability.

➢ Future work directions

➢ Investigate additional fault-tolerance techniques applicable in the FaaS context, such as check-point/restart, logging.

➢ Design a smart, fault-tolerant system for FaaS.