Cold Start Influencing Factors in Function as a Service

Johannes Manner · Martin Endreß · Tobias Heckel · Guido Wirtz  
johannes.manner@uni-bamberg.de

Distributed Systems Group  
Otto-Friedrich-University Bamberg, Germany
Research Questions

- How to benchmark cold starts of cloud functions consistently to get repeatable experiments and results?
- Which factors written down as hypotheses influence the cold start of cloud functions?
Why Investigating Cold Starts?

- In General: First Execution on a newly allocated VM, container etc. is always slower
- Challenges:
  - Performance Unpredictability (No.5)
  - Scaling Quickly (No.8) [Armbrust 2010]
- Cloud Functions are executed in containerized environments

- Overall question: How much overhead does a “simple” cold start face?
- Which factors need deeper insights?
Hypotheses

- H1: Programming Language
- H2: Deployment Package Size
- H3: Memory/CPU Setting
- H4: Number of Dependencies
- H5: Concurrency Level
- H6: Prior Executions
- H7: Container Shutdown

- HX: Discussion during breaks :)}
Experiment – Multiple Executions

- Measurement of the user perception – the user’s cold start
- Network, platform routing etc. is equal for cold and warm starts
  → We only measure the overhead of cold starts
- Having cold and warm executions pairwise since platform shuts down function containers after 20 to 25 minutes (own investigation)
Experiment – Single Execution

Local REST

1. REST Call L-UUID
2. Local Start Timestamp
3. Local End Timestamp

FaaS Platform

4. Platform Start Timestamp
5. Execution – JSON Response L-UUID, C-UUID
6. Platform End Timestamp
7. REST Response L-UUID, C-UUID
8. Local End Timestamp

20.12.2018
Experiment – Data Dimensions

- Experiment was conducted between 6/25/2018 and 7/1/2018
- Each cloud function was invoked 550 times
- 90 deployed functions – 49500 total executions

- H1: Programming Language (Java, JS)
- H2: Deployment Package Size (0, 3, 6, 12, 25, 50, 100, 200, 400 MB)
- H3: Memory/CPU Setting (128, 256, 512, 1024, 2048, 3008 MB)
Experiment – Selected Algorithm

- Recursive Fibonacci

```c
long fibonacci(long n) {
    if (n <= 1) {
        return 1;
    } else {
        return fibonacci(n - 1) + fibonacci(n - 2);
    }
}
```

- Low memory usage $O(n)$
- High CPU usage $O(2^n)$
- Predictable execution time
Where are the cold starts?
Results – Hypotheses Independent

All numbers are mean average values for the execution time in ms

<table>
<thead>
<tr>
<th></th>
<th>Cold</th>
<th>Warm</th>
<th>Cold–Warm</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Java</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Client | 5961 | 4211 | 1750      | 16%
| Platform | 4329 | 4082 | 247       | 63% |
| AWS    |      |      |           |    |
| JS     | 14320| 13676| 644       | 0%  |
| Client |      |      |           |    |
| Platform | 13496| 13539| -43       | 4%  |
| Azure  |      |      |           |    |
| Client | 26681| 1809 | 24872     | 0%  |
| Java   |      |      |           |    |
| Platform | 15261| 1545 | 13716     | 4%  |
| JS     | 14369| 4547 | 9822      | 4%  |
| Client |      |      |           |    |
| Platform | 5492 | 4270 | 1222      | 4%  |
Results – Hypotheses Independent

Client End - Start

Platform End – Start
Results – Hypotheses Dependent

- **H1: Programming Language**
  - Table shows AWS values
  - Azure ratio: 2.53
  - Confirm hypothesis based on the ratios

- **H2: Deployment Package Size**
  - Weak, but present correlation
  → Confirm hypothesis except for Azure Java (no clear tendency)

- **H3: Memory/ CPU Setting**
  - Only quantifiable for AWS
  - Confirm hypothesis
Discussion

Assumption: AWS bills only for function execution

Parameter setting is use case dependent

Available Metrics

User point of view

Avoid ping strategies – Scaling

Platform Limitations

Sample Size

Temporal Relevance

\[
\frac{\text{compiled}}{\text{interpreted}} : 2 \text{ to } 3
\]
Next Steps

- Extending the presented work by including new hypotheses, platforms and dimensions

- Price Analysis based on different load settings (constant load, bursty workloads etc.) and other influential factors. Comparison with other cost benchmarks. Also including the pricing of other ecosystem backend services like databases, notification systems etc.

- Simulation framework for FaaS users to find the best setting for their cloud functions.

I’m very interested to discuss the presented work and the planned research objects to get your feedback during the poster session 😊

Contact

Johannes Manner
/johannes-manner
johannes.manner@uni-bamberg.de